PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-056735 #2
(43)Date of publication of application : 03.03.1995

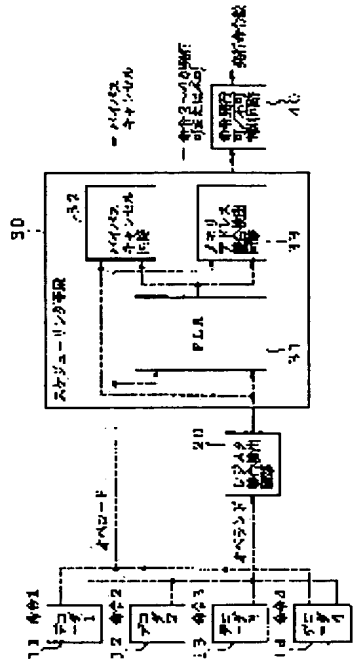
(51)Int.Cl. G06F 9/38
G06F 9/38

(21)Application number : 05-202207
(22)Date of filing : 16.08.1993

(71)Applicant : NKK CORP
(72)Inventor : KATO TAKAAKI
ONO TOSHIHISA

(54) PARALLEL ARITHMETIC AND LOGIC UNIT

(57)Abstract:
PURPOSE: To provide a parallel arithmetic processor allowed to be easily loaded on normal RISC architecture and capable of executing high speed processing for a super scalar without requiring a complicated hardware circuit.
CONSTITUTION: This parallel arithmetic processor is provided with decoders 11 to 14 for simultaneously decoding plural instructions, a register competition detecting circuit 20 for detecting the existence of register competition between two instructions, a PLA 31 including an instruction issue judging logic circuit, an FU use judging logic circuit, an ALU use judging logic circuit, an LD/STU use judging logic circuit, a branch instruction processing logic circuit, an execution order judging logic circuit, etc., a scheduling means 30 including a by-pass canceling circuit 32 and a memory address compentition circuit 33, and an instruction issue enable/disable judging circuit 40 for determining the number of instructions to be issued based upon an instruction issue enabling/disabling information.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

MicroPatent® Worldwide PatSearch: Record 1 of 3

Family of JP7056735A2 How It Works

#2

Family of JP07056735

No additional family members are found for this document

PatentWeb Home | Edit Search | Return to Patent List | Back to Record | Help

MicroPatent® Worldwide PatSearch: Record 1 of 3

Family of JP7056735B4 How It Works

Stage 2 Patent Family - "Extended"		Priorities and Applications	
CC DocNum	KD PubDate	CC AppNum	KD AppDate
☐ JP 7056735	B4 19950614	JP 60485	A 19850325
☐ JP 61219222	A2 19860929	JP 60485	A 19850325
2 Publications found.			

Order Selected Documents

PatentWeb Home | Edit Search | Return to Patent List | Back to Record | Help

MicroPatent® Worldwide PatSearch: Record 1 of 3

Family of JP07056735 How It Works

Your family lookup result have more than 1 family set, please select the patent number to lookup

- (•) JP 7056735 A2 19950303
- ( ) JP 7056735 B4 19950614

Display Family

PatentWeb Home | Edit Search | Return to Patent List | Back to Record | Help

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平7－56735

(43)公開日　平成７年(1995)３月３日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/38	３７０　Ｘ			
	３１０　Ｆ			

審査請求　未請求　請求項の数７　ＯＬ　（全 30 頁）

(21)出願番号　特願平5－202207

(22)出願日　平成５年(1993)８月16日

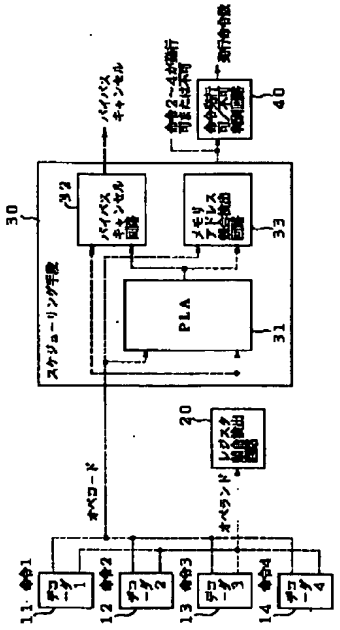
(71)出願人　000004123
日本鋼管株式会社
東京都千代田区丸の内一丁目１番２号
(72)発明者　加藤　高明
東京都千代田区丸の内一丁目１番２号　日本鋼管株式会社内
(72)発明者　小野　利寿
東京都千代田区丸の内一丁目１番２号　日本鋼管株式会社内
(74)代理人　弁理士　佐々木　宗治　（外３名）

(54)【発明の名称】　並列演算処理装置

(57)【要約】

【目的】　スーパースカラを対象として、複雑なハードウエア回路を要さず、通常のＲＩＳＣアーキテクチャ上に容易に実装可能で、しかも従来のＤＳよりも高速処理が可能な並列演算処理装置。

【構成】　複数の命令を同時にデコードするデコーダ１１～１４と、２命令間のレジスタ競合の有無を検出するレジスタ競合検出回路２０と、命令発行判別論理回路、ＦＵ使用判別論理回路、ＡＬＵ使用判別論理回路、ＬＤ／ＳＴＵ使用判別論理回路、分岐命令処理論理回路及び実行順序判別論理回路等を内蔵するＰＬＡ３１、並びにバイパスキャンセル回路３２及びメモリアドレス競合回路３３を含むスケジューリング手段３０と、命令発行の可又は不可情報により発行命令数を決める命令発行可／不可判別回路４０とを備えたもの。



1

【特許請求の範囲】

【請求項１】　複数Ｎ個の命令を同時に解読し、それぞれの命令の種別情報を出力するＮ個のデコーダと、
それぞれ演算処理命令及びロード／ストア命令を実行できる２以上前記Ｎまでの間の任意の数Ｍ個の命令実行ユニット（ＦＵ）と、
前記Ｎ個のデコーダで解読されたＮ個の命令のうちの２個の命令について、これらの命令の間のレジスタの競合の有無を検出し、該検出したレジスタ競合情報を出力するレジスタ競合検出回路と、
前記Ｎ個の命令の種別情報とレジスタ競合情報とを入力して、Ｎ個の命令について同時に実行でき発行可能な命令を判別する命令発行判別論理回路、及びＭ＋１番目以降の各命令について、Ｍ＋１番目までの各命令のうち命令発行可能な命令数が前記Ｍを越えないか、越えるかにより前記命令実行ユニットの使用の可否を判別するＦＵ使用判別論理回路を含むスケジューリング手段とを備えたことを特徴とする並列演算処理装置。

【請求項２】　前記Ｍ個の命令実行ユニット（ＦＵ）に代えて、同時に実行可能な、１以上前記Ｍまでの間の任意の数Ｋ個の演算処理ユニット（ＡＬＵ）と、１以上前記Ｍまでの間の任意の数Ｌ個のロード／ストア処理ユニット（ＬＤ／ＳＴＵ）とを設け、
また前記命令実行ユニットの使用の可否を判別するＦＵ使用判別論理回路に代えて、Ｋ＋１番目以降の各命令について、Ｋ＋１番目までの各命令のうち演算処理命令発行可能な命令数が前記Ｋを越えないか、越えるかにより前記演算処理ユニットの使用の可否を判別するＡＬＵ使用判別論理回路と、Ｌ＋１番目以降の各命令について、Ｌ＋１番目までの各命令のうちロード／ストア命令発行可能な命令数が前記Ｌを越えないか、越えるかにより前記ロード／ストア処理ユニットの使用の可否を判別するＬＤ／ＳＴＵ使用判別論理回路とを設けた請求項１記載の並列演算処理装置。

【請求項３】　前記スケジューリング手段は、前記Ｎ個の命令のうち、Ｊ番目の命令について、Ｊ－１番目までの命令との間にレジスタ競合が無い場合に、Ｊ－１番目までの各命令の実行の可否にかかわらず、Ｊ番目の命令を実行可能と判別し、命令の実行順序を入れ換えて実行させることができる実行順序判別論理回路を含む請求項１または請求項２記載の並列演算処理装置。

【請求項４】　前記スケジューリング手段は、前記Ｎ個の命令のうち、Ｊ番目の命令について、該命令が分岐命令である場合に、Ｊ＋１番目以降の命令を実行不可能と判別し、Ｊ＋１番目以降の命令をスケジューリングの対象からはずす分岐命令処理論理回路を含む請求項１または請求項２記載の並列演算処理装置。

【請求項５】　前記スケジューリング手段は、前記Ｎ個の命令のうち、Ｊ番目の命令について、該命令がロード／ストア命令である場合に、Ｊ－１番目までの各命令の

2

うちでロード／ストア命令であり且つ前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、同一のメモリアドレスにアクセスする命令が無い場合は、Ｊ番目の命令を実行可能と判別し、また同一のメモリアドレスにアクセスする命令が有る場合は、それらがすべてロード命令であり且つＪ番目の命令もロード命令であれば、Ｊ番目の命令を実行可能と判別するメモリ競合判別論理回路を含む請求項１または請求項２記載の並列演算処理装置。

【請求項６】　前記スケジューリング手段は、前記Ｎ個の命令のうち、Ｊ番目以降の命令について、実行可能で且つＪ－１番目以前の命令より２クロック以上先に実行する場合に、該命令の実行順序が逆になることによる競合の有無を判別し、競合が有る場合には、該命令のスケジューリングを１クロック遅らせる遅延指令信号を前記命令発行判別論理回路へ供給する逆順序競合判別論理回路を含み、
且つ前記Ｊ－１番目までの命令のうちで前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、データ読出レジスタがＪ番目以降の各命令のデータ書込レジスタと同一か否かを判別し、同一である場合には、前記命令の実行順番を逆にしてレジスタ競合が有るという情報に基づき、Ｊ番目以降の各命令から該当命令を実行した結果情報のバイパスをキャンセルする信号を出力するバイパスキャンセル論理回路を含む請求項１または請求項２記載の並列演算処理装置。

【請求項７】　前記の各種論理回路は、プログラマブル・ロジック・アレイ（ＰＬＡ）により実現されたものである請求項１乃至請求項６のいずれかに記載の並列演算処理装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明はＲＩＳＣ（Ｒｅｄｕｃｅｄ　Ｉｎｓｔｒｕｃｔｉｏｎ　Ｓｅｔ　Ｃｏｍｐｕｔｅｒ）プロセッサのさらなる高速化を目的としたスーパースカラのための複数命令同時発行機構を有する並列演算処理装置に関するものである。

【０００２】

【従来の技術】図１２は従来のＲＩＳＣアーキテクチャのパイプライン構造を示す図であり、従来のＲＩＳＣアーキテクチャは、単一のプロセッサ（ＣＰＵ）を用いて、図１２に示すようなパイプライン構造による時間的並列処理を導入し、ＣＰＵの命令実行に要する時間の高速化を実現したものである。図１２の各命令は、次の４つのステージの処理を順次行なう。そして各ステージの処理は、それぞれ１クロックサイクルを要するので、各命令の実行開始から終了までには４クロックサイクルを要する。最初に、命令フェッチ（ＩＦ；Ｉｎｓｔｒｕｃｔｉｏｎ　Ｆｅｔｃｈ）ステージの処理を行ない、次に、命令デコード（ＩＤ；Ｉｎｓｔｒｕｃｔｉｏｎ　Ｄ

ecode）ステージの処理を行ない、次に、デコード命令の実行（EX；Execution）ステージの処理を行ない、最後に、実行結果の書き込み（または格納）（WB；Write Back）ステージの処理を行なう。

【0003】図12のパイプライン構造では、各命令は1クロックサイクルずつずらせて時間的に並列処理される。即ち命令1の実行が開始されて1クロックサイクル経過すると、命令2の実行が開始される。そして命令2の実行が開始されて1クロックサイクル経過すると、命令3の実行が開始される。このようにして、RISCでは複数の命令を並列処理できるので、理想的には図12に示すように1クロックサイクルに1命令（CPI＝1；Clock per Instruction）実行することが可能である。しかし実際には、分岐命令やハザード（hazard、詳細は後述する）によるストール（stall；1クロックサイクル以上のパイプラインの停止）があるため、CPIは1以上となる。

【0004】また、コンピュータの命令レベル（Instruction-levelまたはfine-grain）の並列度を利用し、RISCの高速化を目指した並列処理アーキテクチャには、スーパーパイプライン（Superpipelline），スーパースカラ（Superscalar），VLIW（Very Long Instruction Word）がある。RISCの機械語命令（オブジェクト・コード）の互換性があり、並列度の拡張性からスーパースカラが、現時点ではもっとも有望視されている。スーパースカラとは、命令の読み出しから、デコード、実行、結果の格納までの一連の処理を、複数命令に対して、同時に行なうことにより、高速処理を図るプロセッサであると定義される。そして複数命令同時発行方法の中でも、命令のOut-of-order発行は、スーパースカラにもっとも高い性能を与えることから、現在までに多くの研究が行なわれてきた（例えば下記の参考文献のうちの［1～8］を参照）。

【0005】参考文献

［1］R. M. Tomasulo, "An Efficient Algorithm for Exploiting Multiple Arithmetic Units," IBM Journal of Research and Development, Vol. 11, pp. 25-33, 1967.

［2］S. Weiss, and J. E. Smith, "Instruction Issue Logic in Pipelined Supercomputers," IEEE Trans. Comput., Vol. C-33, No. 11, pp. 1013-1022, 1984.

［3］R. D. Acosta, J. Kjelstrup, and H. C. Torng, "An Instruction issuing approach to enhancing performance in multiple functional unit processors," IEEETrans. Comput., Vol. C-35, pp. 815-828, 1986.

［4］W. Hwu, and Y. N. Patt, "HPSm, A highperformance restricted data flow architecture having minimal functionality," Proc. 13th Int'l Symp. Computer Architecture, pp. 297-307, 1986.

［5］G. S. Sohi, and S. Vajapeyam, "Instruction Issue Logic for High-Performance, Interruptible Pipelined Processors," Proc. 14th Int'l Symp. Computer Architecture, pp. 27-34, 1987.

［6］K. Murakami, M. Kuga, and S. Tomita, "SIMP:A Novel High-Speed Single-Processor Architecture," Proc. 16th Int'l Symp. Computer Architecture, pp. 78-85, 1989.

［7］M. D. Smith, M. S. Lam, and M. A. Horowitz, "Boosting Beyond Static Scheduling in a Superscalar Processor," Proc. 17th Int'l Symp. Computer Architecture, pp. 344-353, 1990.

［8］R. M. Keller, "Look-Ahead Processors," Computing Serveys, Vol. 7, No. 4, pp. 175-195, 1975.

［9］M. Johnson, Superscalar Microprocessor Design, Prentice Hall, 1991.

［10］J. E. Smith, and A. R. Pleszkun, "Implementation of Precise Interruts inPipelined Processors," Proc. 12th Int'l Symp. Computer Architecture, pp. 36-44, 1985.

［11］J. L. Hennessy, and D. A. Patterson, "Computer Architecture:A Quantitative Ap

5

proach,"Morgan Kaufmann Publishers, Inc., 1990.

【０００６】ここで命令発行（Instruction issue）とは、命令をパイプラインのデコードステージ（ＩＤ）から実行ステージ（ＥＸ）に動かすプロセスを言う。またＯｕｔ－ｏｆ－ｏｒｄｅｒ発行とは、命令間のデータの依存関係が無く、プログラムが正しく演算されるなら、命令の実行順序を入れ換えて実行する方式である。これに対し、プログラムの命令実行順序で命令を発行する方式をｉｎ－ｏｒｄｅｒ発行という。Ｏｕｔ－ｏｆ－ｏｒｄｅｒ発行の実現方法に関する研究の多くは、前記参考文献の［1，2，4～7］に記載のＴｏｍａｓｕｌｏアルゴリズム及びその拡張法である。しかし、これらの方法では、レジスタ名前替えアルゴリズム（ｒｅｇｉｓｔｅｒ　ｒｅｎａｍｉｎｇ）、予約ステーション（ｒｅｓｅｒｖａｔｉｏｎ　ｓｔａｔｉｏｎ）、タグなどの方法を組み合わせたもので、相対的に複雑なハードウエア回路を必要とし、通常のＲＩＳＣアーキテクチャ上に実装する上で大きな改造を必要とするものである。

【０００７】Ｏｕｔ－ｏｆ－ｏｒｄｅｒ発行を実現する手段には、Ｔｏｍａｓｕｌｏアルゴリズムの他には中央命令ウインドウ方式がある。ここで、命令ウインドウは、フェッチ（ｆｅｔｃｈ；命令の命令キャッシュからの取り込み）された命令を一時的に格納するバッファである。この中央命令ウインドウ方式の実現手段に下記の３方法がある（前記参考文献の［3，7，9］を参照）。

（1）Ｋｅｌｌｅｒの原理（参考文献の［8］）に基づくｄｉｓｐａｔｃｈ　ｓｔａｃｋ法（以下ＤＳと略す、参考文献［3］を参照）

（2）ｒｅｇｉｓｔｅｒ　ｕｐｄａｔｅ　ｕｎｉｔ法（以下ＲＵＵと略す、参考文献の［7］を参照）

（3）リオーダバッファ（ｒｅｏｒｄｅｒ　ｂｕｆｆｅｒ、参考文献の［10］）を使用した方法（参考文献の［9］を参照）

ここで、リオーダバッファとは、分岐予測ミス時のプロセッサの状態回復のためのハードウエアであり、完了した命令による実際のレジスタの値と、未完了命令の保留代入により構成される。上記方法（2）のＲＵＵの性能は方法（1）のＤＳの性能を下回り、同様に方法（3）のリオーダバッファを使用した方法も方法（1）のＤＳの性能を上回ることはない（参考文献の［9］を参照）。

【０００８】

【発明が解決しようとする課題】しかしながら従来の参考文献に示されたようなスーパースカラでは、複雑なハードウエア回路を必要とし、通常のＲＩＳＣアーキテクチャ上に実装するには装置に大きな改造を要するという問題点があった。またＯｕｔ－ｏｆ－ｏｒｄｅｒ発行を

6

実現するＤＳ（ｄｉｓｐａｔｃｈ　ｓｔａｃｋ法）では、レジスタに関する依存関係と、メモリに関するデータ依存関係とを別機構で直列に処理するため、クロックサイクルが長くなり、結果的に処理装置の性能が低下するという問題点があった。

【０００９】本発明はかかる問題点を解決するためになされたもので、スーパースカラを対象として、複雑なハードウエア回路を要さず、通常のＲＩＳＣアーキテクチャ上に容易に実装可能で、しかも従来のＤＳよりも高速処理が可能な並列演算処理装置を得ることを目的とする。

【００１０】

【課題を解決するための手段】本発明の請求項１に係る並列演算処理装置は、複数Ｎ個の命令を同時に解読し、それぞれの命令の種別情報を出力するＮ個のデコーダと、それぞれ演算処理命令及びロード／ストア命令を実行できる２以上前記Ｎまでの間の任意の数Ｍ個の命令実行ユニット（ＦＵ）と、前記Ｎ個のデコーダで解読されたＮ個の命令のうちの２個の命令について、これらの命令の間のレジスタの競合の有無を検出し、該検出したレジスタ競合情報を出力するレジスタ競合検出回路と、前記Ｎ個の命令の種別情報とレジスタ競合情報とを入力して、Ｎ個の命令について同時に実行でき発行可能な命令を判別する命令発行判別論理回路、及びＭ＋１番目以降の各命令について、Ｍ＋１番目までの各命令のうち命令発行可能な命令数が前記Ｍを越えないか、越えるかにより前記命令実行ユニットの使用の可否を判別するＦＵ使用判別論理回路を含むスケジューリング手段とを備えたものである。

【００１１】本発明の請求項２に係る並列演算処理装置は、前記請求項１に係る並列演算処理装置において、前記Ｍ個の命令実行ユニット（ＦＵ）に代えて、同時に実行可能な、１以上前記Ｍまでの間の任意の数Ｋ個の演算処理ユニット（ＡＬＵ）と、１以上前記Ｍまでの間の任意の数Ｌ個のロード／ストア処理ユニット（ＬＤ／ＳＴＵ）とを設け、また前記命令実行ユニットの使用の可否を判別するＦＵ使用判別論理回路に代えて、Ｋ＋１番目以降の各命令について、Ｋ＋１番目までの各命令のうち演算処理命令発行可能な命令数が前記Ｋを越えないか、越えるかにより前記演算処理ユニットの使用の可否を判別するＡＬＵ使用判別論理回路と、Ｌ＋１番目以降の各命令について、Ｌ＋１番目までの各命令のうちロード／ストア命令発行可能な命令数が前記Ｌを越えないか、越えるかにより前記ロード／ストア処理ユニットの使用の可否を判別するＬＤ／ＳＴＵ使用判別論理回路とを設けたものである。

【００１２】本発明の請求項３に係る並列演算処理装置は、前記請求項１または請求項２に係る並列演算処理装置において、前記スケジューリング手段は、前記Ｎ個の命令のうち、Ｊ番目の命令について、Ｊ－１番目までの

7

命令との間にレジスタ競合が無い場合に、Ｊ－１番目までの各命令の実行の可否にかかわらず、Ｊ番目の命令を実行可能と判別し、命令の実行順序を入れ換えて実行させることができる実行順序判別論理回路を含むものである。

【００１３】本発明の請求項４に係る並列演算処理装置は、前記請求項１または請求項２に係る並列演算処理装置において、前記スケジューリング手段は、前記Ｎ個の命令のうち、Ｊ番目の命令について、該命令が分岐命令である場合に、Ｊ＋１番目以降の命令を実行不可能と判別し、Ｊ＋１番目以降の命令をスケジューリングの対象からはずす分岐命令処理論理回路を含むものである。

【００１４】本発明の請求項５に係る並列演算処理装置は、前記請求項１または請求項２に係る並列演算処理装置において、前記スケジューリング手段は、前記Ｎ個の命令のうち、Ｊ番目の命令について、該命令がロード／ストア命令である場合に、Ｊ－１番目までの各命令のうちでロード／ストア命令であり且つ前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、同一のメモリアドレスにアクセスする命令が無い場合は、Ｊ番目の命令を実行可能と判別し、また同一のメモリアドレスにアクセスする命令が有る場合は、それらがすべてロード命令であり且つＪ番目の命令をロード命令であれば、Ｊ番目の命令を実行可能と判別するメモリ競合判別論理回路を含むものである。

【００１５】本発明の請求項６に係る並列演算処理装置は、前記請求項１または請求項２に係る並列演算処理装置において、前記スケジューリング手段は、前記Ｎ個の命令のうち、Ｊ番目以降の命令について、実行可能で且つＪ－１番目以前の命令より２クロック以上先に実行する場合に、該命令の実行順序が逆になることによる競合の有無を判別し、競合が有る場合には、該命令のスケジューリングを１クロック遅らせる遅延指令信号を前記命令発行判別論理回路へ供給する逆順序競合判別論理回路を含み、且つ前記Ｊ－１番目までの命令のうちで前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、データ読出レジスタがＪ番目以降の各命令のデータ書込レジスタと同一か否かを判別し、同一である場合には、前記命令の実行順番を逆にしてレジスタ競合が有るという情報に基づき、Ｊ番目以降の各命令から該当命令を実行した結果情報のバイパスをキャンセルする信号を出力するバイパスキャンセル論理回路を含むものである。

【００１６】本発明の請求項７に係る並列演算処理装置は、前記請求項１乃至請求項６のいずれかに係る並列演算処理装置において、前記の各種論理回路の論理演算を実現したプログラマブル・ロジック・アレイ（ＰＬＡ）を有するものである。

【００１７】

【作用】本請求項１に係る発明においては、Ｎ個のデコ

8

ーダは、複数Ｎ個の命令を同時に解読し、それぞれの命令の種別情報を出力する。２以上前記Ｎまでの間の任意の数Ｍ個の命令実行ユニット（ＦＵ）は、それぞれ演算処理命令及びロード／ストア命令を実行することができる。レジスタ競合検出回路は、前記Ｎ個のデコーダで解読されたＮ個の命令のうちの２個の命令について、これらの命令の間のレジスタの競合の有無を検出し、該検出したレジスタ競合情報を出力する。スケジューリング手段内の命令発行判別論理回路は、前記Ｎ個の命令の種別情報とレジスタ競合情報とを入力して、Ｎ個の命令について同時に実行でき発行可能な命令を判別し、ＦＵ使用判別論理回路は、Ｍ＋１番目以降の各命令について、Ｍ＋１番目までの各命令のうち命令発行可能な命令数が前記Ｍを越えないか、越えるかにより前記命令実行ユニットの使用の可否を判別する。

【００１８】本請求項２に係る発明においては、前記請求項１に係る発明における前記Ｍ個の命令実行ユニット（ＦＵ）に代えて、同時に実行可能な、１以上前記Ｍまでの間の任意の数Ｋ個の演算処理ユニット（ＡＬＵ）と、１以上前記Ｍまでの間の任意の数Ｌ個のロード／ストア処理ユニット（ＬＤ／ＳＴＵ）とが設けられ、また前記命令実行ユニットの使用の可否を判別するＦＵ使用判別論理回路に代えて、ＡＬＵ使用判別論理回路とＬＤ／ＳＴＵ使用判別論理回路とが設けられる。そしてＡＬＵ使用判別論理回路は、Ｋ＋１番目以降の各命令について、Ｋ＋１番目までの各命令のうち演算処理命令発行可能な命令数が前記Ｋを越えないか、越えるかにより前記演算処理ユニットの使用の可否を判別する。またＬＤ／ＳＴＵ使用判別論理回路は、Ｌ＋１番目以降の各命令について、Ｌ＋１番目までの各命令のうちロード／ストア命令発行可能な命令数が前記Ｌを越えないか、越えるかにより前記ロード／ストア処理ユニットの使用の可否を判別する。

【００１９】本請求項３に係る発明においては、前記請求項１または請求項２に係る発明におけるスケジューリング手段に実行順序判別論理回路が含まれ、該実行順序判別論理回路は、前記Ｎ個の命令のうち、Ｊ番目の命令について、Ｊ－１番目までの命令との間にレジスタ競合が無い場合に、Ｊ－１番目までの各命令の実行の可否にかかわらず、Ｊ番目の命令を実行可能と判別し、命令の実行順序を入れ換えて実行させることができる。

【００２０】本請求項４に係る発明においては、前記請求項１または請求項２に係る発明におけるスケジューリング手段に分岐命令処理論理回路が含まれ、該分岐命令処理論理回路は、前記Ｎ個の命令のうち、Ｊ番目の命令について、該命令が分岐命令である場合に、Ｊ＋１番目以降の命令を実行不可能と判別し、Ｊ＋１番目以降の命令をスケジューリングの対象からはずす。

【００２１】本請求項５に係る発明においては、前記請求項１または請求項２に係る発明におけるスケジューリ

9

ング手段にメモリ競合判別論理回路が含まれ、該メモリ競合判別論理回路は、前記N個の命令のうち、J番目の命令について、該命令がロード／ストア命令である場合に、J－1番目までの各命令のうちでロード／ストア命令であり且つ前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、同一のメモリアドレスにアクセスする命令が無い場合は、J番目の命令を実行可能と判別し、また同一のメモリアドレスにアクセスする命令が有る場合は、それらがすべてロード命令であり且つJ番目の命令もロード命令であれば、J番目の命令を実行可能と判別する。

【0022】本請求項6に係る発明においては、前記請求項1または請求項2に係る発明におけるスケジューリング手段に逆順序競合判別論理回路及びバイパスキャンセル論理回路が含まれ、逆順序競合判別論理回路は、前記N個の命令のうち、J番目以降の命令について、実行可能で且つJ－1番目以前の命令より2クロック以上先に実行する場合に、該命令の実行順序が逆になることによる競合の有無を判別し、競合が有る場合には、該命令のスケジューリングを1クロック遅らせる遅延指令信号を前記命令発行判別論理回路へ供給する。またバイパスキャンセル論理回路は、前記J－1番目までの命令のうちで前記レジスタ競合検出回路により競合が検出されて実行ができないとされた命令のうち、データ読出レジスタがJ番目以降の各命令のデータ書込レジスタと同一か否かを判別し、同一である場合には、前記命令の実行順番を逆にしてレジスタ競合が有るという情報に基づき、J番目以降の各命令から該当命令を実行した結果情報のバイパスをキャンセルする信号を出力する。

【0023】本請求項7に係る発明においては、前記請求項1乃至請求項6に係る発明における前記の各種論理回路が、プログラマブル・ロジック・アレイ（PLA）により実現されたものである。

【0024】

【実施例】本発明はスーパースカラを対象としたものであり、同時実行可能な命令の検出と並列実行のためのハードウエア機構が必要になるが、従来の逐次型プロセッサ（スカラプロセッサ）の命令セットを変更せずに、命令レベルの並列処理を達成できるものである。本発明のアルゴリズム、ハードウエア及びPLAの実施例等を説明する前に、本発明の技術概要を従来のDSと比較して説明する。本発明は従来のDSと比較して次の4点に関して優位であるといえる。（なお詳細は後述する）。

（a）DSは、レジスタに関するデータハザードについてのみ解決を図った機構であり、FU（Functional Units；機能ユニットもしくは命令実行ユニット、または1つのパイプライン演算機のことを指す）競合などを回避する場合には新たな機構が必要となる。このためクロック・サイクル・タイムを延ばし、性能低下を引き起こす可能性がある。一般に命令のパイプライン処理を乱す原因は、ハザード（hazard）あるいは競合（dependencyまたはconflict）と呼ばれる。データハザードは命令依存関係のうち、レジスタまたはメモリに格納されたデータ相互間の依存関係のあることを指す。データハザードにはRAW（Read After Write，フロー依存；flow dependence）、WAW（Write After Write，出力依存；output dependence）、WAR（Write After Read，逆依存；antidependence）の各ハザードがある。FU競合（構造ハザード）回避とは、複数の命令が同時に同じFUを使用しないようにすることである。

10

【0025】（b）DSでは、データハザードのうち、レジスタに関する依存関係は命令発行機構（Issue Unit）によって、メモリに関するデータ依存関係は別機構（the Address Stack and the Data Unit）によって、別個に解決される。そのため、クロックサイクルが長くなってしまい、結果的に性能が低くなる。これに対し、本発明による機構は命令発行機構でレジスタ及びメモリの両方に関する依存関係を解決できるものである。DSではメモリに関する依存関係については全く考慮されていないことが性能低下の原因となっている。

（c）本発明はWARハザード回避方法の相違によりDSの性能を上回ることができる。

（d）本発明にはスケーラビリテイ（scalability）がある。ここでスケーラビリテイとは、同一アーキテクチャのもとで要素プロセッサ数をどこまで増加させるかの可能性を意味し、FU数を増減しても対応できる性質と、FUの均質性に依存しないアルゴリズムであるという特性を有する。前記FUの均質性又は不均質性とは、各FUで実行できる命令が同じであるか否かであり、FUが不均質の場合には、実行可能な命令の組み合わせによっては命令クラス・リソース競合（構造ハザード）が生じる。この構造ハザードやメモリに関するデータハザード等を並列的に解決して命令発行を行なう命令発行アルゴリズムを、PLA（Programmable Logic Array）により実現した実施例及びその効果については後述する。

【0026】最初に本発明のアルゴリズムを説明し、その後にハードウエアについての説明を行なう。アルゴリズムについては、以下の説明上の仮定を行なう。

（a）説明上の仮定

（1）説明を簡略化するため、同時に実行可能かどうかを扱う命令数は、4命令を基本とする。4命令はプログラム順にそれぞれ命令1（またはInst. 1，Inst. はInstructionの略とする）、命令2、命令3、命令4と呼ぶ。（なお4命令以上に拡張することは容易である。）

11

（2）全ての命令は1クロックで実行される。そのため、浮動小数点演算やアドレス計算を行なうｌｏａｄ／ｓｔｏｒｅ命令などは考慮しない。ただし、デコード・＊

OP　Rs1，Rs2，Rd：Rd←Rs1　OP　Rs2　……（1）

ここで、OPは命令のオペコード（操作コード、ｏｐｅｒａｔｉｏｎ　ｃｏｄｅ）、Rs1，Rs2はそれぞれソース（ｓｏｕｒｃｅ、出所）レジスタ、Rdはディスティネーション（ｄｅｓｔｉｎａｔｉｏｎ、行先）レジスタ、オペランドはソースレジスタとディスティネーションレジスタからなる。

【0027】（3）パイプライン構造は以下の（イ）～（ニ）の4ステージ構成を想定している。（イ）命令フェッチ（ＩＦ；Ｉｎｓｔｒｕｃｔｉｏｎ　Ｆｅｔｃｈ）ステージ、（ロ）命令デコード（ＩＤ；Ｉｎｓｔｒｕｃｔｉｏｎ　Ｄｅｃｏｄｅ）ステージ、（ハ）デコード命令の実行（ＥＸ；Ｅｘｅｃｕｔｉｏｎ）ステージ、（ニ）実行結果の書き込み（または格納）（ＷＢ；Ｗｒｉｔｅ　Ｂａｃｋ）ステージである。最初に（イ）のＩＦステージでは、4命令を同時に命令キャッシュ（または主記憶装置）から取り込む。4命令の先頭の命令はプログラム・カウンタ（ＰＣ）から与えられる。次の（ロ）のＩＤステージの前半で、4命令のオペコード（加算、減算のような操作を定義するビット列）とオペランド（演算の対象を示す。本例では、具体的にはレジスタの番地を指定する。）がそれぞれデコード（解読）される。ＩＤステージの後半で、オペランドの値がレジ※

例えば、スーパースカラの対ＲＩＳＣ比での性能向上を★

となる。

【0029】図3は本発明とＤＳのＩＤ（命令デコード）ステージにおける処理方式の相違を説明する図であり、ＤＳ方式では、同時にフェッチした複数命令の並列デコード処理の次に、レジスタに関するデータハザードの回避処理、ＦＵ競合の処理、及びメモリに関するデータハザードの回避処理をそれぞれ順番に（時間的には直列処理で）解決せざるを得ない構造となっている。しかし本発明の方式では、上記の3つの処理を同時に（時間的には並列処理で）解決できる構造となっており、レジスタの読み込み処理と後述するスケジューリング処理を並列処理する。このことは、前記本発明と従来のＤＳとの比較した4点において、（a）、（b）項については、クロックサイクル時間ｔを短くすることに相当し、（c）項については、1命令の実行に要するクロックサイクル数ＣＰＩを小さくすることに相当する。

【0030】（b）本発明の命令発行は以下に示す原則に基づいてスケジュールされる。

（1）命令1は常に発行される。

（2）命令n（2から4）が発行される条件は、次のＡ～Ｃの3条件を満たさなければならない。すなわち、3

12

＊ロジックが少し複雑になるが、考慮することは容易である。命令形式は次の（1）式の通りである。

※スタ・ファイルから読み込まれる。この読み込みと並行して命令発行のためのスケジューリングがなされる。次の（ハ）のＥＸステージではＡＬＵ（Ａｒｉｔｈｍｅｔｉｃ　ｕｎｉｔ、演算処理ユニット）命令は演算を行ない、ｌｏａｄ／ｓｔｏｒｅ命令はデータ・キャッシュ・メモリ（Ｄａｔａ　Ｃａｃｈｅ　Ｍｅｍｏｒｙ）にアクセスする。最後の（ニ）のＷＢステージでは、ＥＸステージの実行結果をレジスタ・ファイルに書き込む。本発明に係る上記4ステージ構成は、5段を有するパイプライン（例えば前記参考文献の［11］等）のほとんどのＲＩＳＣプロセッサに適用が可能である。

【0028】（4）ＣＰＵの性能（プログラムの実行に要するＣＰＵ時間）及び性能向上については以下の（2）式のように表せられる。

プログラムの実行に要するＣＰＵ時間
＝プログラムの終了に要するクロック数×ｔ
＝ＩＣ×ＣＰＩ×ｔ　……（2）

ここで、ＩＣはプログラム内の命令数、ＣＰＩは1命令の実行に要するクロックサイクル数、ｔはクロックサイクル時間である。性能向上（ｓｐｅｅｄｕｐ）はアムダール（Ａｍｄａｈｌ）の法則（前記参考文献の［11］参照）により、次の（3）式のように示される。

性能向上＝比較対象の実行時間／被比較対象の実行時間　……（3）

★見る場合は、

ＲＩＳＣの実行時間／スーパースカラの実行時間　……（3′）

条件のＯＲ｛ＲＡＷ＋ＷＡＷ＋ＷＡＲ＋分岐＋ＦＵ競合；＋は論理和（ＯＲ）を示すものとする。これらの用語の説明は後述｝が成立するときに発行される。

Ａ．（ＲＡＷ、ＷＡＷ、ＷＡＲ条件）
命令（ｎ－1）以前の命令と命令ｎとの間でデータ競合（ＲＡＷ、ＷＡＷ、及びＷＡＲハザード）が無いこと。

Ｂ．（分岐条件）
命令（ｎ－1）以前の命令が分岐命令で無いこと。

Ｃ．（ＦＵ競合条件）
命令ｎが発行されると仮定して、命令ｎまでの命令のうち、発行される命令数が機能ユニットの数を越えないこと。

【0031】Ｄ．本機構では、上記の競合条件を、各命令間の実行時間の差を記述したスケジューリング・パターンを用いて、あらかじめ解決しておくものである。このスケジューリング・パターンは、ウインドウ内の命令が発行可能であるかを記述したもので、想定され得る入力パターン（レジスタ競合及び4命令の命令種別）に対し、論理演算を並列化及び圧縮化して、ＰＬＡから出力されるべきスケジューリング・パターンが実装される。これによってハードウエア回路を簡易化し、かつ実行ク

13

ロックサイクルの短縮化を図ったものである。命令１は必ず発行されるので、命令２～４がそれぞれ発行可能であるか否かを示す３ｂｉｔデータがＰＬＡの出力となる。この最終パターンが決定するまでのスケジューリング・パターンは、各命令がそれぞれのハザードを考慮すると何クロックサイクル目で実行可能となるかを絶えず考慮している。命令１の場合を実行クロック・サイクル“１”と呼ぶ。実行クロック・サイクル“１”は言い換えると、スケジューリングの結果命令発行可能ということである。１クロックサイクル分ストールすると、実行クロック・サイクルは“２”となる。実行時にはデコード情報（レジスタ競合情報である１３ｂｉｔ及び４命令の命令種別情報である８ｂｉｔのデータ）がＰＬＡに入力され、パターン・マッチングによってスケジューリング・パターンが出力される。

【００３２】（３）本発明のパイプライン構造におけるレジスタに関するデータハザードの回避方法について述べる。

Ａ．ＲＡＷ（Ｒｅａｄ　Ａｆｔｅｒ　Ｗｒｉｔｅ）ハザード

命令ｎが命令（ｎ－１）以前の命令の結果を使って演算する命令である場合、命令ｎは命令（ｎ－１）以前の命令の実行後に実行されなければならない。そのため、命令ｎは命令（ｎ－１）以前の命令とは同時に実行することはできない。これをＲＡＷハザードと呼ぶ。図４は本発明に係るバイパス回路によるＲＡＷハザード回避方法の説明図である。本アルゴリズムではＲＡＷハザードが検出されると、命令ｎは発行されない。従って、図４のように１クロック分だけ命令の発行が待たされる。図４において、Ｉｎｓｔ．２の発行が１クロック分待たされても、Ｉｎｓｔ．１の結果がクロック４の前半でレジスタ・ファイルに書き込まれるのに対し、Ｉｎｓｔ．２はクロック３の後半でレジスタからデータをすでに読み込んでしまっており、依然としてＲＡＷハザードが生じている。この問題は、ＦＵにバイパス（ｂｙｐａｓｓ；短絡路）をつけて、出力を入力へフィードバックするフォワーデイング（ｆｏｒｗａｒｄｉｎｇ）技術によって解決される（前記参考文献の［１１］を参照）。

【００３３】Ｂ．ＷＡＷ（Ｗｒｉｔｅ　Ａｆｔｅｒ　Ｗｒｉｔｅ）ハザード

命令ｎの実行結果の格納場所（レジスタまたはメモリ）が、命令（ｎ－１）以前の命令の実行結果の格納場所と同じ場合、命令ｎの実行結果の書き込みが命令（ｎ－１）以前の命令の結果の書き込みより早くなされると、命令（ｎ＋１）以降の命令は間違った値を得てしまう。そこで命令ｎは命令（ｎ－１）以前の命令とは同時に実行することはできない。これをＷＡＷハザードと呼ぶ。図５は本発明に係るＷＡＷハザード回避方法の説明図である。本アルゴリズムではＷＡＷハザードが検出されると、命令ｎは発行せず、ＲＡＷハザード回避の方法と同様にして１クロック分発行が待たされる。図５の例では、命令１～３はすべて加算命令であり、命令３は命令１及び２とＲＡＷハザードは無いが、命令３と命令１のＲｄが共通のＲ３であり、命令１とＷＡＷハザードがあるため、命令３の実行は１クロック分遅らされる。このような例が生じるのは、命令１のＲｄと命令２のＲｓ１が共通のＲ３であって、命令２が命令１の結果を使用するためである。

14

【００３４】Ｃ．ＷＡＲ（Ｗｒｉｔｅ　Ａｆｔｅｒ　Ｒｅａｄ）ハザード

命令（ｎ－１）以前の命令が、レジスタまたはメモリの値を読み込む前に、命令ｎがレジスタまたはメモリに実行結果を書き込んでしまうと、命令（ｎ＋１）以降の命令は間違った値を得てしまう。これをＷＡＲハザードと呼ぶ。図６は本発明に係るＷＡＲハザード現象の説明図である。図６の例では、命令２は命令１と、命令３は命令２とそれぞれＲＡＷハザードを持つが、命令４は命令１，２，３のいずれともＲＡＷ及びＷＡＷハザードを持たないため、命令４はＯｕｔ－ｏｆ－ｏｒｄｅｒ効果によって命令３より実行が２クロック早まる。ところが、命令４のＲｄと命令３のＲｓ２が一致するため、ＷＡＲハザードとなる。しかし、１クロックのみ早まるだけでは、ＷＡＲハザードが生じないのが、図６からわかる。

【００３５】本発明による複数命令同時発行方式は、スケジューリング・パターン方式と称して命令ウインドウ内の４命令の実行完了がそれぞれ何クロック目でなされるかを考慮して命令発行を決定する方式である。２クロック早まるのは、図６の例すなわち、命令３・４間でしか起こり得ない。図６において命令１～４までの実行クロックサイクルは、それぞれ３，４，５，３である。命令１の場合を実行クロック・サイクル“１”と呼ぶと、命令２～４は“２，３，１”となり、ＷＡＲハザードが生じるのは、実行クロック・サイクルが（１２３１）というスケジュール・パターンに限定されることがわかる。本アルゴリズムでは、スケジュール・パターンが（１２３１）で、かつ命令４のＲｄと命令３のＲｓ間のレジスタ競合があるときにＷＡＲハザードとみなす。そして、これが検出されるとスケジューリング・パターン上で命令４の発行が１クロック遅らされる。

【００３６】スケジューリング・パターンにおいて、命令１は常に発行されるから、実行クロック・サイクルは“１”である。命令２は、命令１とＲＡＷ及びＷＡＷハザードを持たなければ、命令１と同時に実行可能であり、実行クロック・サイクルは“１”だが、いずれかのハザードが存在すれば、実行クロック・サイクルは“２”となる。命令３は、命令１・２のスケジューリング・パターンが（１，１）の時は“１”または“２”で、命令１・２のスケジューリング・パターンが（１，２）の時は“１”、“２”または“３”ということになる。命令ｎの実行クロック・サイクルは“１～ｎ”であ

15

り、スケジューリング・パターンが（１１１３）のようにクロックサイクル“2”である命令が無いというような不連続なパターンは無い。但し、Ｏｕｔ－ｏｆ－ｏｒｄｅｒの効果により（１２３１）というパターンは存在する。スケジュールの対象になる命令（命令ウインドウ＊

16

＊内の命令数）が４命令の場合のスケジューリング・パターンは１５通りであり、これを次の表１に示す。

【００３７】

【表１】

４命令のスケジューリング・パターン

	SP	N1_4		SP	N1_4
1)	1111	4	9)	1221	2
2)	1112	3	10)	1222	1
3)	1121	3	11)	1223	1
4)	1122	2	12)	1231	2
5)	1123	2	13)	1232	1
6)	1211	3	14)	1233	1
7)	1212	2	15)	1234	1
8)	1213	2			

SP: スケジューリング・パターン
N1_4: 発行可能な命令数

【００３８】（４）分岐命令の扱いについて

本方式では、命令（ｎ－１）が分岐命令であったら、命令ｎ以降の命令は取り除いてスケジューリングをする（即ち命令ｎ以降の発行及び実行をしない）。これは、本方式がＯｕｔ－ｏｆ－ｏｒｄｅｒ命令発行方式なので、分岐命令以降の命令が分岐命令よりも先に実行されてしまうと、間違った答が得られてしまうことが頻繁に起こるからである。これについて図７を用いて説明する。図７は本発明に係る分岐命令を飛び越えてＯｕｔ－ｏｆ－ｏｒｄｅｒが実行される場合の説明図である。図７の例では、命令２は命令１とＲＡＷハザードを持ち、命令３は命令２の結果を使用している（ＲＡＷハザード）。命令３は分岐命令（ＢＲＴ）で、命令２の実行結果をクロックサイクル４において分岐先命令のアドレスが確定し、クロックサイクル５で始めて命令フェッチが可能である。命令３は命令２の結果が格納されるレジスタＲ５の値で真偽が判断され、真であれば、次の命令である命令４からの４つの命令が、偽であれば、命令Ｘからの４つの命令がフェッチされるという命令である。命令４は分岐命令の後続命令であり、命令１～３と同時にスケジューリングされる。このとき、命令４は命令１～３とＲＡＷ，ＷＡＷ，ＷＡＲハザードを持たないために、Ｏｕｔ－ｏｆ－ｏｒｄｅｒ効果によって命令３の分岐命令より前に実行される。命令３の実行結果から分岐先命令として命令Ｘが選択された場合、すでに命令４は実行されており、命令４の結果がレジスタＲ１０に書き込まれていて、レジスタＲ１０を参照する命令Ｘは間違った答を得てしまう。

【００３９】しかし、本方式では、スケジューリング時に分岐命令の後続の命令は無視するため、スケジューリングの対象にならず（命令ウインドウ内には残らず）、分岐命令の結果が決定するまで命令発行はされない。従って、スケジューリング・パターン上では分岐命令の後続の命令は無意味なものとして処理され、次項で述べるＦＵ競合やメモリに関するデータ競合（メモリアドレス競合）については対象外となる。なお、多分岐を実現するには複雑なハードウエアを必要とすること、及び多分岐を扱うプロセッサは稀であるので、本方式では対象としなかった。また、スケジューリングの対象となる命令の中で、分岐命令が検出されると、後続の命令はスケジューリングから外されるため、スケジューリング対象命令の中に分岐命令が複数あると、自動的に最初の分岐命令だけが有効となる。これに対して、分岐予測という方法がある。この方法は、上記の例において分岐予測を行ない、命令４を選択した場合に、予測した結果が正しい場合には、実行時間の大幅短縮になる。しかし予測がミスした場合には、予測した命令の実行を無効化し、パイプラインが予測前の正しいプロセッサの状態を回復する処置を要する方式である。しかし、この復元に要するハードウエア機構はかなり複雑であるため、本方式では採

17

用しないこととした。

【００４０】（５）ＦＵ競合の回避方法について

ＦＵ競合（構造ハザード）とは、命令間にデータハザードが無く発行可能であったとしても、それを実行する適切なＦＵがなければ実行できないということで、これを回避しなければならない。このＦＵ競合は、ＲＩＳＣ、スーパーパイプライン、ＶＬＩＷでは起こり得ず、スーパースカラ特有のことである。スーパースカラでもスケジュールの対象命令数とＦＵの数が等しければＦＵ競合は発生しない。命令実行ユニットが（ｎ－１）個なら、スケジューリングの対象になる命令（命令ウインドウ内の命令数）も（ｎ－１）個で良いように思えるが、（ｎ－１）個の命令が必ずしもすべて実行可能ではない。そのためいくつかの命令実行ユニットは未使用状態で、効率が良くない。そこで、スケジューリング対象命令数は、命令実行ユニット数より多いのが一般的である。この場合、ｎ個以上の命令が同時に実行できることがあり、命令を実行できるユニットが無いため、命令ｎは発行することができないことになる。

18

【００４１】本発明においては、均質（Ｕｎｉｆｏｒｍ）ユニット構成はＡＬＵ命令とＬｏａｄ／Ｓｔｏｒｅ命令を実行でき、不均質（ｎｏｎ－ｕｎｉｆｏｒｍ）ユニット構成ではＡＬＵ命令もしくはＬｏａｄ／Ｓｔｏｒｅ命令を実行できるものと想定した。いずれの場合も、その他に分岐命令の命令実行ユニットを１個有する。通常のＲＩＳＣあるいはスカラプロセッサはＡＬＵ、Ｌｏａｄ／Ｓｔｏｒｅ、分岐の全ての命令実行ユニットを持っているが、常時このうち１ユニットのみが実行されると定義できる。命令ウインドウ内の命令数が４であれば、ＦＵユニット構成のタイプは、均質タイプがＡ～Ｃの３種、不均質タイプがＤ～Ｓの１６種で合計１９種となり、これを次の表２に示す。

【００４２】

【表２】

19　　20

命令数が4の時の機能ユニット構成

タイプ（均質）	Nn		Nb	FU競合の有無	Ss
A	4		1	X	4
B	3		1	O	3
C	2		1	O	2
不均質	Nalu	Nl/s	Nb	FU競合の有無	Ss
D	1	1	1	O	3
E	2	1	1	O	4
F	3	1	1	O	4
G	4	1	1	O	4
H	1	2	1	O	4
I	1	3	1	O	4
J	1	4	1	O	4
K	2	2	1	O	4
L	2	3	1	O	4
M	2	4	1	O	4
N	3	2	1	O	4
O	4	2	1	O	4
P	3	3	1	O	4
Q	3	4	1	O	4
R	4	3	1	O	4
S	4	4	1	X	4

Ss:並列発行度またはスーパースカラ度

O：考慮の必要あり、x:考慮の必要無し

Nn:均質ユニット構成におけるＦＵ数

Nalu:不均質ユニット構成における　ＡＬＵ数

Nl/s:不均質ユニット構成における
　　Ｌｏａｄ／Ｓｔｏｒｅユニット数

Nb:　両ユニット構成における分岐ユニット数

【００４３】ユニット構成が均質の場合及び不均質の場合におけるＦＵ競合が存在する条件は、それぞれ次の＊

ｉｆ（Ｎｎ＜N_{1-4}），→ｔｈｅｎ　ＦＵ競合　……（4）

ｉｆ｛（Ｎａｌｕ＜N_{1-4}）＊（Ｎｌ／ｓ＜N_{1-4}）｝，
→ｔｈｅｎ　ＦＵ競合　……（5）

即ちユニット構成が均質の場合には（4）式の左側の条件が満足された場合に、またユニット構成が不均質の場合には（5）式の左側が条件が満足された場合に、それぞれＦＵ競合が存在する。ここで、Ｎｎは均質ユニット構成の時のＦＵ数、Ｎａｌｕは不均質ユニット構成の時のＡＬＵの数、Ｎｌ／ｓは不均質ユニット構成の時のＬｏａｄ／Ｓｔｏｒｅユニットの数、＊は論理積（ＡＮＤ）を示す。

【００４４】表２に示されるタイプが１９種あるユニット構成のうち、ＦＵ競合の有無の欄が“Ｘ”で示されるタイプＡとＳはＦＵ競合が起きない。これは命令数に対してＦＵの数が十分にあるからである。しかし、この欄

21

が“O”で示される他のユニット構成については、式（4）及び（5）を基に、ＦＵ競合を考慮しなければならない。各スケジューリング・パターンとN_{1-4}との関係は表１に示される。

【００４５】均質ユニット構成の場合、ＮｎとＮ$_{1-4}$を比較することによって、スケジューリング・パターンは式（4）の該当部において変えられなければならない。一方、不均質ユニット構成の場合、ＮａｌｕまたはＮ１／ｓの数とＮ$_{1-4}$を比較することになるが、この場合は単純な比較ではなく、Ｎ$_{1-4}$のうち何命令がＡＬＵ命令であるか、何命令がＬｏａｄ／Ｓｔｏｒｅ命令であるかが問われる。すなわち、データハザード及び分岐命令の有無の結果、発行可能な命令のうちＡＬＵ命令（もしく＊はＬｏａｄ／Ｓｔｏｒｅ命令）の数が、Ｎａｌｕ（もしくはＮ１／ｓ）より小さい場合は命令発行をしないようにしなければならない。これは、命令の種別に依存し、かつ実行時でないとわからない。具体的には、命令ｎのＦＵ競合回避は、命令（ｎ－１）までの命令が発行可能かどうかを積算しなければならない。これをそのままハードウエア回路で実現すると、直列的な処理となるため演算に時間がかかり実行クロックサイクルが長くなるので、本発明ではこれをあらかじめ論理展開し、並列演算可能になるようにしている。ＦＵ競合を考慮した結果のスケジューリング・パターンを次の表３に示す。

22

【００４６】

【表３】

ＦＵ競合を考慮したスケジューリング・パターン

	SP	SP1					SP	SP1	
1)	1111	1111	1211	1121	1112	8)	1213	1213	1223
		1221	1212	1122	1222	9)	1221	1221	1222
2)	1112	1112	1212	1121	1222	10)	1222	1222	
3)	1121	1121	1221	1122	1222	11)	1223	1223	
4)	1122	1122	1222			12)	1231	1231	1232
5)	1123	1123	1223			13)	1232	1232	
6)	1211	1211	1212	1221	1222	14)	1233	1233	
7)	1212	1212	1222			15)	1234	1234	

SP1: ＦＵ競合を考慮したスケジューリング・パターン
○：FU 競合回避の結果、ＷＡＲ ハザードを引き起こす可能性のあるＳＰ（１２３１）に変わり得るＳＰ１

【００４７】コンパイル時にＦＵの競合が存在するかどうかを検出しようとすると、大幅に並列度が損なわれることになる。例えば、スケジューリング・パターンが（１１１２）でタイプＥ（ＡＬＵユニットが２でＬｏａｄ／Ｓｔｏｒｅユニットが１）の場合を考えてみる。この場合Ｎ$_{1-4}$のうちＡＬＵが３命令以上もしくはＬｏａｄ／Ｓｔｏｒｅ命令が２命令以上であると命令３は発行できなくなるが、ＡＬＵが２命令で、Ｌｏａｄ／Ｓｔｏｒｅ命令が１命令であれば、実行可能となる。これを判別するためには、実行時に４命令の命令種別を見て、かつこれらのＮａｌｕまたはＮ１／ｓの数とＮ$_{1-4}$を比較しなければならない。

【００４８】（6）ＦＵ競合回避によって生ずるＷＡＲハザードの回避

ＦＵ競合を考慮した結果、スケジューリング・パターンが（１２３１）となってしまい、ＷＡＲハザードを引き起こす可能性が生じる。例えばスケジューリング・パターンが（１１２１）であったとする。またタイプＤ（ＡＬＵユニットが１でＬｏａｄ／Ｓｔｏｒｅユニットが１）の場合で、実行時の命令種別が命令１と２がＡＬＵ命令、命令４がＬｏａｄ／Ｓｔｏｒｅ命令であったとする。この場合、命令２はＦＵ競合により発行できず、スケジューリング・パターンは（１２３１）に変えられなければならない。本方式では、ＦＵ競合の結果（１２３１）となったパターンについては再度ＷＡＲハザードの有無を見ている。ＦＵ競合の結果（１２２１）となるパターンが次のスケジューリング時に再びＦＵ競合を起こし、（１２３１）パターンとなる。この例を表３に図示した。表３において、ＳＰ１はＦＵ競合を考慮したスケジューリング・パターンであり、このなかで楕円で囲まれたものは、ＦＵ競合回避の結果、ＷＡＲハザードを引き起こす可能性のあるＳＰ（１２３１）に変わり得るＳＰ１である。これまでスケジューリング・パターンは実行サイクルクロック数を考慮してきたが、ハードウエア

23　　24

回路として必要なのは発行できるか（１）否か（０）である。ここで出力されるスケジューリング・パターンは、次の表４のＳＰ２に示すようなスケジューリング・＊パターンがＰＬＡの出力となる。

【００４９】

【表４】

4命令の出力 スケジューリング・パターン

	SP	SP2	in-order/ out-of-order		SP	SP2	in-order/ out-of-order
1)	1111	1111	in-order	9)	1221	1001	out-of-order
2)	1112	1110	in-order	10)	1222	1000	in-order
3)	1121	1101	out-of-order	11)	1223	1000	in-order
4)	1122	1100	in-order	12)	1231	1001	out-of-order
5)	1123	1100	in-order	13)	1232	1000	in-order
6)	1211	1011	out-of-order	14)	1233	1000	in-order
7)	1212	1010	out-of-order	15)	1234	1000	in-order
8)	1213	1010	out-of-order				

SP2: 出力スケジューリング・パターン　（１：命令発行する、　０：命令発行せず）

【００５０】表４において、ＳＰ２の欄は４命令の出力スケジューリング・パターンを示し、欄内の（１）は命令を発行できる場合を、（０）は命令を発行できない場合をそれぞれ示している。ここで、注意を要することがある。図６の例においてレジスタに関するＷＡＲハザードは命令４が２クロックではなく、１クロックのみ早く実行されることによって回避される。しかし、１クロック早く実行されるとＲＡＷハザードとみなされ、バイパス回路が開かれＷＡＲハザードと同じ現象となり、命令３は間違った答を得てしまう。このようなＷＡＲハザード条件はスケジューリング・パターン上で回避できるが、この場合は、バイパス回路を開かないようにして並列度を損なうことなく回避することができ、これをバイパス・キャンセル方式と呼ぶ。Ｏｕｔ－ｏｆ－ｏｒｄｅｒ発行された場合で、かつＯｕｔ－ｏｆ－ｏｒｄｅｒになった命令間で次のクロックに命令の逆順によるＲＡＷハザードを引き起こすことになるレジスタ競合がある場合に、バイパス回路はキャンセルされなければならない。この条件を次の表５に示す。

30

【００５１】

【表５】

WAR ハザード回避のための バイパス回路 をキャンセルする条件

	SP2	条件	閉じられるバイパス回路
1)	1111	条件無し	
2)	1110	条件無し	
3)	1101	AND (rc5 OR rc6)	命令３のＲｓ１またはＲｓ２
4)	1100	条件無し	
5)	1100	条件無し	
6)	1011	AND (rc1 OR rc2)	命令２のＲｓ１またはＲｓ２
	1011	AND (rc3 OR rc4)	命令２のＲｓ１またはＲｓ２
	1011	AND (rc1 OR rc2) AND (rc3 OR rc4)	命令２のＲｓ１またはＲｓ２ 命令３のＲｓ１またはＲｓ２
7)	1010	AND (rc1 OR rc2)	命令２のＲｓ１またはＲｓ２
8)	1010	AND (rc1 OR rc2)	命令２のＲｓ１またはＲｓ２
9)	1001	AND (rc3 OR rc4)	命令２のＲｓ１またはＲｓ２
	1001	AND (rc5 OR rc6)	命令３のＲｓ１またはＲｓ２
10)	1000	条件無し	
11)	1000	条件無し	
12)	1001	AND (rc3 OR rc4)	命令２のＲｓ１またはＲｓ２
	1001	AND (rc5 OR rc6)	命令３のＲｓ１またはＲｓ２
13)	1000	条件無し	
14)	1000	条件無し	
15)	1000	条件無し	

rc1: 命令３のＲｄと命令２のＲｓ１間のレジスタ競合
rc2: 命令３のＲｄと命令２のＲｓ２間のレジスタ競合
rc3: 命令４のＲｄと命令２のＲｓ１間のレジスタ競合
rc4: 命令４のＲｄと命令２のＲｓ２間のレジスタ競合
rc5: 命令４のＲｄと命令３のＲｓ１間のレジスタ競合
rc6: 命令４のＲｄと命令３のＲｓ２間のレジスタ競合

【００５２】表５のＳＰ２の欄における数字の“１”は命令を実行する場合を、“０”は命令を実行しない場合をそれぞれ示している。これに対し、ＤＳ方式では各命令が何クロック目で実行されるかということには着目せず、ＲＡＷ、ＷＡＷ及びＷＡＲハザードを引き起こす可能性のあるレジスタ競合の論理和（ＯＲ）を取り、これらのレジスタ競合が一つもなければ命令が発行される。この方法の良さはレジスタに関するデータハザードだけに関して言えば、ハードウエア機構が単純化されることである。しかし、性能に関して言えば、本発明のスケジューリング・パターン方式の方がＤＳ方式より良い。これは、上記に述べたようにＷＡＲハザードを引き起こす可能性のあるレジスタ競合が存在しても、後続の命令が１クロックのみ実行が早まるだけではＷＡＲハザードは生じないということによる。この典型例を図８に示す。

【００５３】図８は本発明に係るＷＡＲハザード回避方法及びその効果の説明図である。図８の例では、命令２が命令１とＲＡＷハザードを持つため実行が１クロック遅らされる。命令３は命令１，２とＲＡＷ及びＷＡＷハザードを持たないが、ＷＡＲハザードを引き起こす可能性のあるレジスタ競合が存在する。この場合に本発明の方式では、命令３は命令２に比べて１クロックのみ実行が早まるので、実際はＷＡＲハザードではないが、ＤＳ方式では各命令の実行クロック数を見ていないために、この例のようにＷＡＲハザードを引き起こす可能性のあるレジスタ競合が生じる限り、ＷＡＲハザードとみなさざるを得ない。本方式はＤＳ方式に比べ図８の（ａ），（ｂ）に示したように命令の実行時間が短縮される。

27

【００５４】（７）メモリに関するデータハザード（メモリアドレス競合）の回避方法について
マルチポート・キャッシュ・メモリを使用すれば、命令ウインドウ内の複数のＬｏａｄ／Ｓｔｏｒｅ命令が同時実行可能である。しかし、これらの命令がＯｕｔ－ｏｆ－ｏｒｄｅｒ発行されると、レジスタに関するデータハザードが無くても、メモリに関するデータハザードを引き起こすことがある。このメモリに関するデータハザードは、これらの命令が参照するレジスタは異なっても、レジスタが指し示すアドレスが一致することにより生じ、命令の実行時でないとわからない。これには次の３つのケースが存在する。
・Ｌｏａｄ命令の後のＳｔｏｒｅ命令がＯｕｔ－ｏｆ－ｏｒｄｅｒ発行されると、ＷＡＲハザードが起きる可能性がある。
・Ｓｔｏｒｅ命令の後のＬｏａｄ命令がＯｕｔ－ｏｆ－

28

ｏｒｄｅｒ発行されると、ＲＡＷハザードが起きる可能性がある。
・Ｓｔｏｒｅ命令の後のＳｔｏｒｅ命令がＯｕｔ－ｏｆ－ｏｒｄｅｒ発行されると、ＷＡＷハザードが起きる可能性がある。
それゆえ、２命令以上のＬｏａｄ／Ｓｔｏｒｅ命令がＯｕｔ－ｏｆ－ｏｒｄｅｒ発行される場合、命令発行される命令がＬｏａｄ命令だけに限られる場合を除いて、Ｉｎ－ｏｒｄｅｒ発行に変えられなければならない。この場合に対象となるのは、表４の（ＳＰ２）に示す出力スケジューリング・パターンのうちＯｕｔ－ｏｆ－ｏｒｄｅｒ発行となるものであり、実行時に命令２～４の命令種別に基づいてハードウエア回路によって変えられる。変えられるべき条件を次の表６の（ＳＰ３）に示す。
【００５５】
【表６】

29　　30

メモリに関するデータハザードを考慮した
スケジューリング・パターン

	SP2	変更されるべきLoad/Store 命令の組合せ条件	SP3
1)	1111	変更無し	1111
2)	1110	変更無し	1110
3)	1101	(y0 AND z1) OR (y1 AND z0) OR (y1 AND.z1)	1100
4)	1100	変更無し	1100
5)	1100	変更無し	1100
6)	1011	(x0 AND y1) OR (x1 AND y0) OR (x1 AND y1)	1001
	1011	(x0 AND y1) OR (x1 AND y0) OR (x1 AND y1)	1010
	1011	(x0 AND y1 AND z0) OR (x0 AND y1 AND z1) OR (x1 AND y0 AND z1) OR (x1 AND y1 AND z0) OR (x1 AND y1 AND z1)	1000
7)	1010	(x0 AND y1) OR (x1 AND y0) OR (x1 AND y1)	1000
8)	1010	(x0 AND y1) OR (x1 AND y0) OR (x1 AND y1)	1000
9)	1001	(x0 AND z1) OR (x1 AND z0) OR (x1 AND z1)	1000
	1001	(x0 AND y1) OR (x1 AND y0) OR (x1 AND y1)	1000
10)	1000	変更無し	1000
11)	1000	変更無し	1000
12)	1001	(x0 AND z1) OR (x1 AND z0) OR (x1 AND z1)	1000
	1001	(x0 AND y1) OR (x1 AND y0) OR (x1 AND y1)	1000
13)	1000	変更無し	1000
14)	1000	変更無し	1000
15)	1000	変更無し	1000

SP3: メモリに関するデータハザードを考慮した変更後の
スケジューリング・パターン

x0: Inst.2 = Load,　x1: Inst.2 = Store,　y0: Inst.3 = Load,　y1: Inst.3 = Store,

z0: Inst.4 = Load,　z1: Inst.4 = Store,

【００５６】次に本発明の複数命令同時発行機構について説明する。

（ａ）ハードウエア

本発明のハードウエア構成を図面を用いて説明する。図２は本発明に係るスーパースカラ・アーキテクチャを示すブロック図である。同図において、１は命令デコード及びスケジューリング機構、２はプログラム・カウンタ（ＰＣ）及び分岐機構、３は命令キャッシュ及びフェッチ機構、４は機能ユニット（ＦＵ）、５はデータキャッシュ、６はレジスタ・ファイルであり、本アーキテクチャは主に上記の６つの部分から構成される。均質ユニット構成の場合、各機能ユニットはＡＬＵ命令とＬｏａｄ／Ｓｔｏｒｅ命令を実行することができる。不均質ユニット構成の場合、機能ユニットはＡＬＵ命令もしくはＬｏａｄ／Ｓｔｏｒｅ命令を実行することができる。Ｌｏａｄ／Ｓｔｏｒｅユニットだけがデータキャッシュにアクセスされる。均質又は不均質ユニットのいずれの構成においても分岐命令は分岐機構で処理される。

【００５７】図１は図２の命令デコード及びスケジューリング機構についての詳細な構成図である。本機構は４つのデコーダ１１～１４、レジスタ競合検出回路２０、スケジューリング手段３０及び命令発行可／否判別回路

31

４０から構成される。またスケジューリング手段３０には、ＰＬＡ３１、バイパスキャンセル回路３２及びメモリアドレス競合検出回路３３が含まれている。ＰＬＡ３１には、複数の命令のうち同時に実行でき発行可能な命令を判別する命令発行判別論理回路、複数のＦＵの使用の可否を判別するＦＵ使用判別論理回路、複数のＡＬＵの使用の可否を判別するＡＬＵ使用判別論理回路、複数のロード／ストア処理ユニット（ＬＤ／ＳＴＵ）の使用の可否を判別するＬＤ／ＳＴＵ使用判別論理回路、分岐命令以降の命令をスケジューリングの対象からはずす分岐命令処理論理回路、命令の実行順序を入れ換えて実行できるか否かを判別する実行順序判別論理回路などの各種論理回路が含れている。

【００５８】また図１においては、バイパスキャンセル回路３２とメモリアドレス競合検出回路３３は、比較的まとまりやすい回路なので、独立の回路構成とした例を示したが、これらの回路をＰＬＡ３１に含ませた構成とすることも可能である。本発明では、このように命令発行に係る各種論理回路をＰＬＡ（Ｐｒｏｇｒａｍｍａｂｌｅ　Ｌｏｇｉｃ　Ａｒｒａｙ）のようなハードウェア装置を用いて実現されるため、ＣＰＵの数を増やすなどの装置の構成が変更されても、ＰＬＡ内のアルゴリズムをソフトウエア的に変更するだけで、他のハードウエア要素をいっさい変えずに済むという特徴がある。バイパスキャンセル回路３２は、命令の実行順序を逆にしてレジスタ競合が有る場合に、該当命令を実行した結果情報のバイパスをキャンセルする回路である。命令発行可／不可判別回路４０は、スケジューリング手段３０内のＰＬＡ３１及びメモリアドレス競合検出回路３３を介して、出力される命令２～４の発行可または発行不可の判別信号により発行命令数を決める回路である。

【００５９】本発明の命令デコード及びスケジューリングの操作は、次の７段階に分かれる。

（１）ウインドウ内の４命令は、それぞれ同時に独立にデコードされ、オペコード及びオペランドが解読される。４命令のオペランドを比較することによって命令間のレジスタ競合が１３ｂｉｔのデータに基づきレジスタ競合検出回路２０で検出される。オペコードのデコード結果は、４種類の命令（ＡＬＵ、Ｌｏａｄ、Ｓｔｏｒｅ、そして分岐命令の４種類）に分類される。各命令は２ｂｉｔで記述され、合計８ｂｉｔである。

（２）デコードが完了すると、ソースレジスタの値がレジスタ・ファイルから読み込まれる。レジスタ読み込みと並行してスケジューリングが以下のようになされる（図３参照）。

（３）レジスタ競合と命令種別の情報（前記１３ｂｉｔと８ｂｉｔの合計２１ｂｉｔのデータ）がＰＬＡに入力される。分岐命令の有無やウインドウ内での位置は、命令種別データより同定される。パターン・マッチングの結果、ＰＬＡからスケジューリング・パターンが出力される。この出力は命令２～４がそれぞれ発行可能であるか否かを示すもので、３ｂｉｔのデータである。このスケジューリング・パターンと命令種別とが前記表６の条件を満足する場合、メモリアドレス競合とみなされ、該当する命令の発行は取り止められる。表６の論理を実現したものが図１のメモリアドレス競合検出回路３３である。

32

【００６０】（４）データパス選択部で、ＦＵのディスティネーションレジスタと発行される命令のソースレジスタとの間にＲＡＷハザードが検出されると、レジスタ・ファイルから読み出された値の代わりにバイパスされた値を選択する（［００３２］の（３）Ａ項［ＲＡＷハザード］を参照）。

（５）発行可能な命令のオペコードとソースレジスタとディスティネーションレジスタのそれぞれの値が適切なＦＵに送られる。

（６）ＷＡＲハザードを引き起こす可能性のあるレジスタ競合とスケジューリング・パターンとの組合せが検出されたなら、バイパス回路キャンセル回路３２においてバイパス回路は不能にされる。

（７）ウインドウから実際に発行された命令の数だけ、スケジューリングのため次の命令をフェッチしなければならない。次にフェッチされる命令のアドレスを保持する。ＰＣは直前に発行された命令数に依存して決定されるため、スケジューリング・パターンからどの命令が発行されたかに基づいて算出される。

【００６１】（ｂ）スケジューリング・パターン及びＰＬＡの設計

スケジューリング・パターンはＰＬＡのハードウエア・サイズが小さくなるように設計される。ＰＬＡへの入力は前記２１ｂｉｔであり、その内訳は命令間のレジスタ競合に１３ｂｉｔ、命令種別の解読に８ｂｉｔである。これらの詳細を以下に示す。

（１）命令間のレジスタ競合（１３ｂｉｔ）

・ＲＡＷ　ｈａｚａｒｄに関するレジスタ競合（６ｂｉｔ）；ｅ，ｆ，ｇ，ｈ，ｉ，ｊ

・ＷＡＷ　ｈａｚａｒｄに関するレジスタ競合（６ｂｉｔ）；ｋ，ｌ，ｍ，ｎ，ｏ，ｐ

・ＷＡＲ　ｈａｚａｒｄに関するレジスタ競合（１ｂｉｔ）；ｑ

ここで、ｅ：命令１・２間のＲｄ－Ｒｓレジスタ競合（ＲＡＷ）、

ｆ：命令１・３間のＲｄ－Ｒｓレジスタ競合（ＲＡＷ）、

ｇ：命令２・３間のＲｄ－Ｒｓレジスタ競合（ＲＡＷ）、

ｈ：命令１・４間のＲｄ－Ｒｓレジスタ競合（ＲＡＷ）、

ｉ：命令２・４間のＲｄ－Ｒｓレジスタ競合（ＲＡＷ）、

33

ｊ：命令３・４間のＲｄ－Ｒｓレジスタ競合（ＲＡＷ）、
ｋ：命令１・２間のＲｄ－Ｒｄレジスタ競合（ＷＡＷ）、
ｌ：命令１・３間のＲｄ－Ｒｄレジスタ競合（ＷＡＷ）、
ｍ：命令２・３間のＲｄ－Ｒｄレジスタ競合（ＷＡＷ）、
ｎ：命令１・４間のＲｄ－Ｒｄレジスタ競合（ＷＡＷ）、
ｏ：命令２・４間のＲｄ－Ｒｄレジスタ競合（ＷＡＷ）、
ｐ：命令３・４間のＲｄ－Ｒｄレジスタ競合（ＷＡＷ）、
ｑ：命令４・３間のＲｓ－Ｒｄレジスタ競合（ＷＡＲ）である。

【００６２】（２）命令解読（８ｂｉｔ）

各命令はそれぞれ次の２ｂｉｔで記述され、合計８ｂｉｔである。

ＡＬＵ命令　　……１０、
ＬＯＡＤ命令　……００、
ＳＴＯＲＥ命令　……０１、
ＢＲＡＮＣＨ命令……１１

命令１～４までのデコード結果（各２ｂｉｔ）を次のように表記する。

命令１の上位ビット…ｄ１１、命令１の下位ビット…ｄ１０、
命令２の上位ビット…ｄ２１、命令２の下位ビット…ｄ２０、

34

命令３の上位ビット…ｄ３１、命令３の下位ビット…ｄ３０、
命令４の上位ビット…ｄ４１、命令４の下位ビット…ｄ４０、

【００６３】（３）レジスタに関するデータハザードの回避手順

Ａ．上記の１３ｂｉｔのレジスタ競合データを基にレジスタに関するデータハザードを考慮してスケジューリング・パターンを作成する。この結果のスケジューリング・パターンが前記表１に集約される。

Ｂ．分岐命令を考慮してスケジューリング・パターンが変更される。具体的には命令１が分岐命令であれば、命令２以降の命令が、命令２が分岐命令であれば、命令３以降の命令が、命令３が分岐命令であれば、命令４がそれぞれ自動的に発行不可とされる。

Ｃ．ＦＵ競合を考慮することによって表３のＳＰ１のようなスケジューリング・パターンに変更される。

Ｄ．ＦＵ競合回避によって生じるＷＡＲハザードを表３の楕円印に着目して回避して、スケジューリング・パターンはＰＬＡの出力として表４のＳＰ２のような形でまとめられる。

Ｅ．これらの論理を圧縮し、共通項を抽出することによって実際のＰＬＡに搭載される積項（ｐｒｏｄｕｃｔ　ｔｅｒｍｓ）は次の表７に示すように１９～４３ｂｉｔでかなり小さくなり、妥当なハードウエア・サイズと言える。

【００６４】

【表７】

35　　36

各ユニット構成におけるＰＬＡ上の積項数

タイプ	均質	命令2 RAW WAW	命令2 分岐	命令2 FUC	命令2 WAR	命令2 小計	命令3 RAW WAW	命令3 分岐	命令3 FUC	命令3 WAR	命令3 小計	命令4 RAW WAW	命令4 分岐	命令4 FUC	命令4 WAR	命令4 小計	合計
A		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	X(0)	O(4)	11	19
B		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	O(1)	O(4)	12	20
C		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(2)	O(4)	13	22
	不均質																
D		O(2)	O(1)	O(2)	X(0)	5	O(4)	O(1)	O(4)	X(0)	9	O(6)	O(1)	O(6)	O(16)	29	43
E		O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(3)	X(0)	8	O(6)	O(1)	O(5)	O(9)	21	33
F		O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(4)	O(9)	20	31
G		O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(3)	O(9)	19	30
H		O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(3)	X(0)	8	O(6)	O(1)	O(5)	O(9)	21	33
I		O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(4)	O(9)	20	31
J		O(2)	O(1)	O(1)	X(0)	4	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(3)	O(9)	19	30
K		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(2)	X(0)	7	O(6)	O(1)	O(4)	O(4)	15	25
L		O(2)	O(1)	O(1)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(3)	O(4)	14	23
M		O(2)	O(1)	O(1)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(2)	O(4)	13	22
N		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(3)	O(4)	14	23
O		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	O(1)	X(0)	6	O(6)	O(1)	O(2)	O(4)	13	22
P		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	O(2)	O(4)	12	21
Q		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	O(1)	O(4)	12	20
R		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	O(1)	O(4)	12	20
S		O(2)	O(1)	X(0)	X(0)	3	O(4)	O(1)	X(0)	X(0)	5	O(6)	O(1)	X(0)	O(4)	11	19

O: 考慮の必要あり、 X: 考慮の必要無し、( ): 積項の数、 FUC FU競合

【００６５】表７において、ユニット構成によって積項が異なるのはＦＵ競合回避方法が異なるためである。また表７に示すように積項の数を減少させることによって、本スケジューリング・パターン方式が小さなハードウエア構成で実現でき、ＦＵユニット構成が変わっても、ＰＬＡ内のアルゴリズムをソフトウエア的に変更するだけで、他のハードウエアはいっさい変えることなく、複数命令をＯｕｔ－ｏｆ－ｏｒｄｅｒで同時に発行できることがわかる。

【００６６】本発明のＰＬＡの一実施例を図面を用いて説明する。図９～図１１は、それぞれユニット構成がタイプA、B、H（表２参照）の場合のＰＬＡ上の命令発行論理回路図である。

ＰＬＡの実施例１

タイプA（均質ユニットが４つ）の場合の実施例

図９はタイプAのユニット構成の場合のＰＬＡ上の命令発行論理回路図である。まず、命令２の発行に支障があるかどうかは図９の（a）の回路により、命令１のディスティネーションレジスタと命令２のソースレジスタが同一であるか（e）、命令１と命令２のディスティネーションレジスタが同一であるか（k）、または、命令１が分岐命令（ｄ１１＝１，ｄ１０＝１，従ってｄ１１＊ｄ１０＝１）の場合に、図９の（a）の回路の出力が１となり発行に支障がある。それ以外では、出力は０となり支障無しとなり発行される。

【００６７】次に、命令３の発行に支障があるかどうかは図９の（b）の回路により、命令１のディスティネーションレジスタと命令３のソースレジスタが同一であるか（f）、命令２のディスティネーションレジスタと命令３のソースレジスタが同一であるか（g）、命令１と命令３のディスティネーションレジスタが同一であるか（l）、命令２と命令３のディスティネーションレジス

タが同一であるか（m）、または、命令１か命令２が分岐命令（ｄ１１＊ｄ１０＝１またはｄ２１＊ｄ２０＝１）の場合に、図９の（b）の回路の出力が１となり発行に支障がある。それ以外では、出力は０となり支障無しとなり発行される。

【００６８】最後に、命令４の発行に支障があるかどうかは図９の（c）の回路により、命令１のディスティネーションレジスタと命令４のソースレジスタが同一であるか（h）、命令２のディスティネーションレジスタと命令４のソースレジスタが同一であるか（i）、命令３のディスティネーションレジスタと命令４のソースレジスタが同一であるか（j）、命令１と命令４のディスティネーションレジスタが同一であるか（n）、命令２と命令４のディスティネーションレジスタが同一であるか（o）、命令３と命令４のディスティネーションレジスタが同一であるか（p）、または、命令１か命令２か命令３が分岐命令（ｄ１１＊ｄ１０＝１またはｄ２１＊ｄ２０＝１またはｄ３１＊ｄ３０＝１）であるか、または、命令１のディスティネーションレジスタと命令２のソースレジスタが同一であるか（e）、命令１と命令２のディスティネーションレジスタが同一であり（k）、かつ命令２のディスティネーションレジスタと命令３のソースレジスタが同一であるか（g）、命令２と命令３のディスティネーションレジスタが同一であり（m）、かつ命令４のディスティネーションレジスタと命令３のソースレジスタが同一である場合（q）に、図９の（c）の回路の出力が１となり発行に支障がある。それ以外では、出力は０となり支障無しとなり発行される。

【００６９】ＰＬＡの実施例２

タイプＢ（均質ユニットが３つ）の場合の実施例

図１０はタイプＢのユニット構成の場合のＰＬＡ上の命令発行論理回路図である。まず、命令２の発行に支障があるかどうかは図１０の（a）の回路により、命令１のディスティネーションレジスタと命令２のソースレジスタが同一であるか（e）、命令１と命令２のディスティネーションレジスタが同一であるか（k）、または、命令１が分岐命令（ｄ１１＝１，ｄ１０＝１，従ってｄ１１＊ｄ１０＝１）の場合に、図１０の（a）の回路の出力が１となり発行に支障がある。それ以外では、出力は０となり支障無しとなり発行される。

【００７０】次に、命令３の発行に支障があるかどうかは図１０の（b）の回路により、命令１のディスティネーションレジスタと命令３のソースレジスタが同一であるか（f）、命令２のディスティネーションレジスタと命令３のソースレジスタが同一であるか（g）、命令１と命令３のディスティネーションレジスタが同一であるか（l）、命令２と命令３のディスティネーションレジスタが同一であるか（m）、または、命令１か命令２が分岐命令（ｄ１１＊ｄ１０＝１またはｄ２１＊ｄ２０＝１）の場合に、図１０の（b）の回路の出力が１となり

発行に支障がある。それ以外では、出力は０となり支障無しとなり発行される。

【００７１】最後に、命令４の発行に支障があるかどうかは図１０の（c）の回路により、命令１のディスティネーションレジスタと命令４のソースレジスタが同一であるか（h）、命令２のディスティネーションレジスタと命令４のソースレジスタが同一であるか（i）、命令３のディスティネーションレジスタと命令４のソースレジスタが同一であるか（j）、命令１と命令４のディスティネーションレジスタが同一であるか（n）、命令２と命令４のディスティネーションレジスタが同一であるか（o）、命令３と命令４のディスティネーションレジスタが同一であるか（p）、または、命令１か命令２か命令３が分岐命令（ｄ１１＊ｄ１０＝１またはｄ２１＊ｄ２０＝１またはｄ３１＊ｄ３０＝１）であるか、

【００７２】または、命令１のディスティネーションレジスタと命令２のソースレジスタが同一で無く（ｌｅ、ｅの論理反転）、かつ命令１と命令２のディスティネーションレジスタが同一で無く（ｌｋ）、かつ命令１のディスティネーションレジスタと命令３のソースレジスタが同一で無く（ｌｆ）、かつ命令２のディスティネーションレジスタと命令３のソースレジスタが同一で無く（ｌｇ）、かつ命令１と命令３のディスティネーションレジスタが同一で無く（ｌｌ）、かつ命令２と命令３のディスティネーションレジスタが同一で無いか（ｌｍ）、または、命令１のディスティネーションレジスタと命令２のソースレジスタが同一であるか（e）、命令１と命令２のディスティネーションレジスタが同一であり（k）かつ命令２のディスティネーションレジスタと命令３のソースレジシタが同一であるか（g）、命令２と命令３のディスティネーションレジスタが同一であり（m）かつ命令４のディスティネーションレジスタと命令３のソースレジスタが同一である場合（q）に、図１０の（c）の回路の出力が１となり発行に支障がある。それ以外では、出力は０となり支障無しとなり発行される。

【００７３】ＰＬＡの実施例３

タイプＨ（不均質ユニット構成、ＡＬＵが１つとＬｏａｄ／ｓｔｏｒｅユニットが２つ）の場合の実施例

図１１はタイプＨのユニット構成の場合のＰＬＡ上の命令発行論理回路図である。まず、命令２の発行に支障があるかどうかは図１１の（a）の回路により、命令１のディスティネーションレジスタと命令２のソースレジスタが同一であるか（e）、命令１と命令２のディスティネーションレジスタが同一であるか（k）、または、命令１が分岐命令（ｄ１１＝１，ｄ１０＝１）であるか（ｄ１１＊ｄ１０＝１）、または、命令１がＡＬＵ命令（ｄ１１＝１，ｄ１０＝０）でかつ命令２もＡＬＵ命令（ｄ２１＝１，ｄ２０＝０）の場合に、図１１の（a）の回路の出力が１となり発行に支障がある。それ以外で

は、出力は０となり支障無しとなり発行される。

【００７４】次に、命令３の発行に支障があるかどうかは図１１の（ｂ）の回路により、命令１のディスティネーションレジスタと命令３のソースレジスタが同一であるか（ｆ）、命令２のディスティネーションレジスタと命令３のソースレジスタが同一であるか（ｇ）、命令１と命令３のディスティネーションレジスタが同一であるか（ｌ）、命令２と命令３のディスティネーションレジスタが同一であるか（ｍ）、または、命令１か命令２が分岐命令（ｄ１１＊ｄ１０＝１またはｄ２１＊ｄ２０＝１）であるか、または、命令１がＡＬＵ命令（ｄ１１＝１，ｄ１０＝０）でかつ命令３もＡＬＵ命令（ｄ３１＝１，ｄ３０＝０）であるか、または、命令２がＡＬＵ命令（ｄ２１＝１，ｄ２０＝０）であってかつ命令３もＡＬＵ命令（ｄ３１＝１，ｄ３０＝０）でありかつ、命令１のディスティネーションレジスタと命令２のソースレジスタが同一で無く（！ｅ）、かつ命令１と命令２のディスティネーションレジスタが同一で無いか（！ｋ）、または、命令１がＬｏａｄ／ｓｔｏｒｅ命令で（ｄ１１＝０）かつ、命令２がＬｏａｄ／ｓｔｏｒｅ命令で（ｄ２１＝０）かつ、命令３がＬｏａｄ／ｓｔｏｒｅ命令で（ｄ３１＝０）かつ、命令１のディスティネーションレジスタと命令２のソースレジスタが同一で無く（！ｅ）、かつ命令１と命令２のディスティネーションレジスタが同一で無い（！ｋ）場合に、図１１の（ｂ）の回路の出力が１となり発行に支障がある。それ以外では、出力は０となり支障無しとなり発行される。

【００７５】最後に、命令４の発行に支障があるかどうかは図１１の（ｃ）の回路により、命令１のディスティネーションレジスタと命令４のソースレジスタが同一であるか（ｈ）、命令２のディスティネーションレジスタと命令４のソースレジスタが同一であるか（ｉ）、命令３のディスティネーションレジスタと命令４のソースレジスタが同一であるか（ｊ）、命令１と命令４のディスティネーションレジスタが同一であるか（ｎ）、命令２と命令４のディスティネーションレジスタが同一であるか（ｏ）、命令３と命令４のディスティネーションレジスタが同一であるか（ｐ）、または、命令１か命令２か命令３が分岐命令（ｄ１１＊ｄ１０＝１またはｄ２１＊ｄ２０＝１またはｄ３１＊ｄ３０＝１）であるか、または、命令１がＡＬＵ命令（ｄ１１＝１，ｄ１０＝０）でかつ命令４もＡＬＵ命令（ｄ４１＝１，ｄ４０＝０）であるか、または、命令２がＡＬＵ命令（ｄ２１＝１，ｄ２０＝０）であってかつ命令４もＡＬＵ命令（ｄ４１＝１，ｄ４０＝０）でありかつ、命令１のディスティネーションレジスタと命令２のソースレジスタが同一で無く（！ｅ）、かつ命令１と命令２のディスティネーションレジスタが同一で無いか（！ｋ）、

【００７６】または、命令３がＡＬＵ命令（ｄ３１＝１，ｄ３０＝０）であってかつ命令４もＡＬＵ命令（ｄ４１＝１，ｄ４０＝０）でありかつ、命令１のディスティネーションレジスタと命令３のソースレジスタが同一で無く（！ｆ）、かつ命令２のディスティネーションレジスタと命令３のソースレジスタが同一で無く（！ｇ）、かつ命令１と命令３のディスティネーションレジスタが同一で無く（！ｌ）、かつ命令２と命令３のディスティネーションレジスタが同一で無いか（！ｍ）、または、命令２がＬｏａｄ／ｓｔｏｒｅ命令で（ｄ２１＝０）かつ、命令４がＬｏａｄ／ｓｔｏｒｅ命令で（ｄ４１＝０）かつ、命令１のディスティネーションレジスタと命令２のソースレジスタが同一で無く（！ｅ）、かつ命令１と命令２のディスティネーションレジスタが同一で無いか（！ｋ）、または、命令１がＬｏａｄ／ｓｔｏｒｃ命令で（ｄ１１＝０）かつ、命令３がＬｏａｄ／ｓｔｏｒｅ命令で（ｄ３１＝０）かつ、命令４がＬｏａｄ／ｓｔｏｒｅ命令で（ｄ４１＝０）かつ、命令１のディスティネーションレジスタと命令３のソースレジスタが同一で無く（！ｆ）、かつ命令２のディスティネーションレジスタと命令３のソースレジスタが同一で無く（！ｇ）、かつ命令１と命令３のディスティネーションレジスタが同一で無く（！ｌ）、かつ命令２と命令３のディスティネーションレジスタが同一で無いか（！ｍ）、または、下記の信号（イ）、及び（ロ）が共に１で、かつ命令４のディスティネーションレジスタと命令３のソースレジスタが同一である場合（ｑ）に、図１１の（ｃ）の回路の出力が１となり発行に支障がある。それ以外では、出力は０となり支障無しとなり発行される。

【００７７】但し、信号（イ）は、命令１のディスティネーションレジスタと命令２のソースレジスタが同一であるか（ｅ）、命令１と命令２のディスティネーションレジスタが同一であるか（ｋ）、または命令１がＡＬＵ命令（ｄ１１＝１，ｄ１０＝０）でかつ命令２もＡＬＵ命令（ｄ２１＝１，ｄ２０＝０）である場合に、１となる。また信号（ロ）は、命令２のディスティネーションレジスタと命令３のソースレジスタが同一であるか（ｇ）、命令２と命令３のディスティネーションレジスタが同一であるか（ｍ）、または命令２がＡＬＵ命令（ｄ２１＝１，ｄ２０＝０）でかつ命令３もＡＬＵ命令（ｄ３１＝１，ｄ３０＝０）である場合に、１となる。

【００７８】なお、本発明は、スーパースカラを対象とするものではあるが、必ずしもこれに限定されるものではなく、パイプライン化された演算機が複数あれば、スーパースカラに限らずスーパーコンピュータなどにも適用可能である。また、対象はＲＩＳＣに限定されず、ＣＩＳＣ（Ｃｏｍｐｌｅｘ　ＩｎｓｔｒｕｃｔｉｏｎＳｅｔ　Ｃｏｍｐｕｔｅｒ）プロセッサに対しても実装は容易である。

【００７９】

【発明の効果】以上のように本発明によれば、複数Ｎ個の命令を同時に解読し、それぞれの命令の種別情報を出

41

力するＮ個のデコーダと、それぞれ演算処理命令及びロード／ストア命令を実行できる２以上前記Ｎまでの間の任意の数Ｍ個の命令実行ユニット（ＦＵ）と、前記Ｎ個のデコーダで解読されたＮ個の命令のうちの２個の命令間のレジスタ競合情報を出力するレジスタ競合検出回路と、Ｎ個の命令について同時に実行でき発行可能な命令を判別する命令発行判別論理回路、及びＭ＋１番目以降の各命令について、前記命令実行ユニットの使用の可否を判別するＦＵ使用判別論理回路を含むスケジューリング手段とを備えるようにしたので、ＲＩＳＣなどのスカラプロセッサのオブジェクト・コード（機械語命令）の互換性を維持しつつ、コンパイラなどのソフトウエアの変更をいっさい行なわずに、通常のＲＩＳＣアーキテクチャ上に実装可能となり、複数命令を同時に発行して実行することにより、ＲＩＳＣなどのスカラプロセッサの高速化・高性能化が可能となる。

【００８０】また本発明によれば、前記Ｍ個の命令実行ユニット（ＦＵ）に代えて、同時に実行可能な、１以上前記Ｍまでの間の任意の数Ｋ個の演算処理ユニット（ＡＬＵ）と、１以上前記Ｍまでの間の任意の数Ｌ個のロード／ストア処理ユニット（ＬＤ／ＳＴＵ）とを設け、また前記ＦＵ使用判別論理回路に代えて、Ｋ＋１番目以降の各命令について、前記演算処理ユニットの使用の可否を判別するＡＬＵ使用判別論理回路と、Ｌ＋１番目以降の各命令について、前記ロード／ストア処理ユニットの使用の可否を判別するＬＤ／ＳＴＵ使用判別論理回路とを設けるようにしたので、表２で示した不均質ユニット構成の場合にも、ＦＵ競合の有無を判別し、複数命令を同時に発行して実行することにより、同様にスカラプロセッサの高速化が可能となる。

【００８１】また本発明によれば、スケジューリング手段内に、命令の実行順序を入れ換えて実行の可否を判別する実行順序判別論理回路を設けるようにしたので、命令間のデータの依存関係が無く、プログラムが正しく演算され得る場合には、命令の実行順序を入れ換えて実行するＯｕｔ－ｏｆ－ｏｒｄｅｒを実施することが可能となり、スカラプロセッサの性能が大幅に向上した。

【００８２】また本発明によれば、スケジューリング手段内に、分岐命令を識別すると、それ以降の命令をスケジューリングの対象からはずす分岐命令処理判別論理回路を設けるようにしたので、従来の分岐予測法における予測ミスの場合の復元に要する複雑なハードウエア機構は不要となり、ハードウエアが簡略化された。

【００８３】また本発明によれば、スケジューリング手段内に、命令がロード／ストア命令である場合にメモリ競合を判別するメモリ競合判別論理回路を設けるようにしたので、パイプライン内の命令の流れを乱す前記レジスタに関するデータハザード及びＦＵ競合と共に、メモリに関するデータハザードなど各種ハザードが並列的に解決されて、スケジューリングが行なわれるので、命令

42

の実行クロックサイクルの短縮化が計られ高速化が可能となる。

【００８４】また本発明によれば、スケジューリング手段内に、命令の実行順序を逆にすることによる競合の有無を判別し、競合が有る場合にスケジューリングを遅らせる逆順序競合判別論理回路と、前記逆順序により競合が有るという情報に基づき該当命令を実行した結果情報のバイパスをキャンセルするバイパスキャンセル論理回路とを設けるようにしたので、誤った演算結果情報を得ることなく、Ｏｕｔ－ｏｆ－ｏｒｄｅｒで複数命令を同時発行する装置を実現することができる。

【００８５】また本発明によれば、前記パイプライン内の命令の流れを乱す各種ハザードを解決し、命令スケジューリングの対象となる命令間の実行されるべき実行クロック時間の差に着目したスケジューリング・パターン方式により作成した命令発行アルゴリズムを論理圧縮してＰＬＡに実装することによって、Ｏｕｔ－ｏｆ－ｏｒｄｅｒで複数命令の同時発行を行うハードウエア機構を簡易化できる。また、命令実行時には命令間のレジスタ競合データと命令種別データとがＰＬＡに入力されると、前記命令発行アルゴリズムによって各命令の発行の可又は不可情報が出力され、パイプラインのクロックサイクルが短縮化されるので、演算処理の高速化が実現されている。さらに、機能ユニットを増やすなどユニット構成が変更されても、ＰＬＡ内のアルゴリズムをソフトウエア的に変更するだけで、他のハードウエア要素をいっさい変更しないでよいという利点がある。

【図面の簡単な説明】

【図１】図２の命令デコード及びスケジューリング機構についての詳細な構成図である。

【図２】本発明に係るスーパースカラ・アーキテクチャを示すブロック図である。

【図３】本発明とＤＳのＩＤステージにおける処理方式の相違を説明する図である。

【図４】本発明に係るバイパス回路によるＲＡＷハザード回避方法の説明図である。

【図５】本発明に係るＷＡＷハザード回避方法の説明図である。

【図６】本発明に係るＷＡＲハザード現象の説明図である。

【図７】本発明に係る分岐命令を飛び越えてＯｕｔ－ｏｆ－ｏｒｄｅｒが実行される場合の説明図である。

【図８】本発明に係るＷＡＲハザード回避方法及びその効果の説明図である。

【図９】タイプＡのユニット構成の場合のＰＬＡ上の命令発行論理回路図である。

【図１０】タイプＢのユニット構成の場合のＰＬＡ上の命令発行論理回路図である。

【図１１】タイプＨのユニット構成の場合のＰＬＡ上の命令発行論理回路図である。

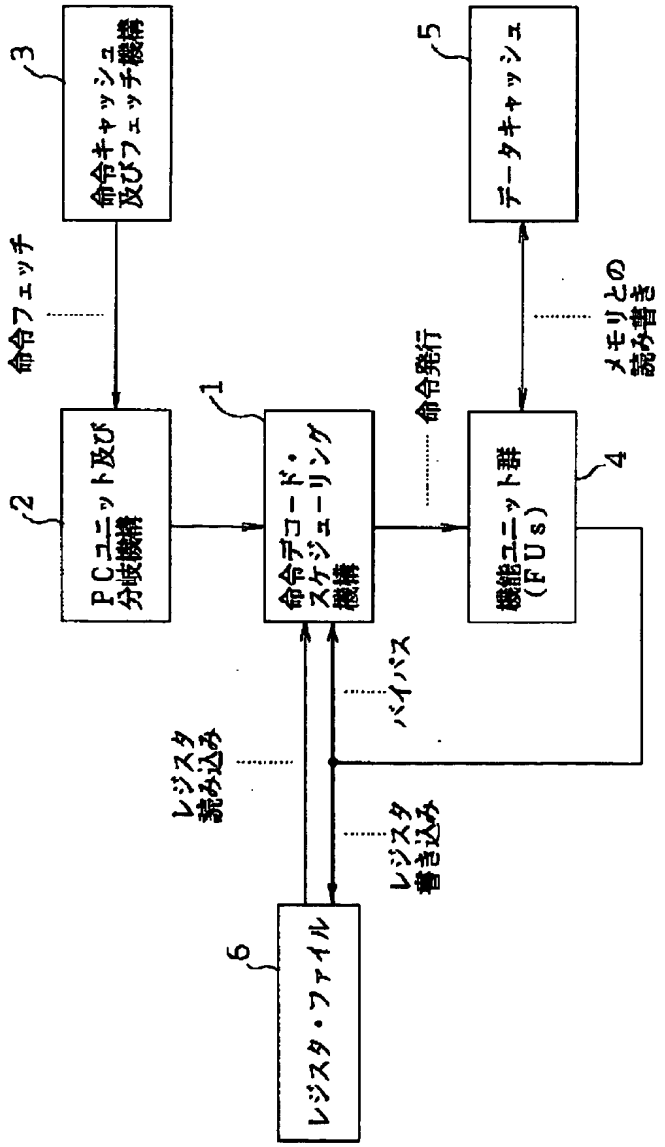
43　　44

【図１２】従来のＲＩＳＣアーキテクチャのパイプライン構造を示す図である。

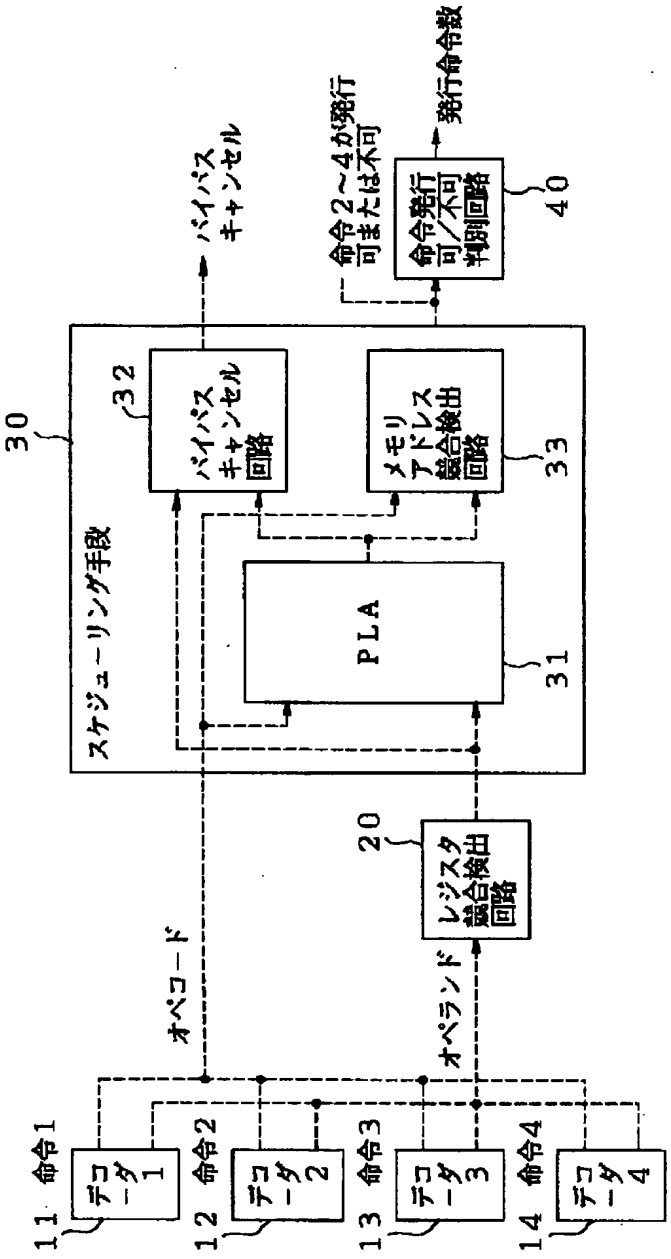
【符号の説明】

1　命令デコード・スケジューリング機構
2　ＰＣユニット及び分岐機構
3　命令キャッシュ及びフェッチ機構
4　機能ユニット群（ＦＵｓ）
5　データキャッシュ
6　レジスタ・ファイル
１１－４　デコーダ１～デコーダ４
２０　レジスタ競合検出回路
３０　スケジューリング手段
３１　ＰＬＡ
３２　バイパスキャンセル回路
３３　メモリアドレス競合検出回路
４０　命令発行可／不可判別回路

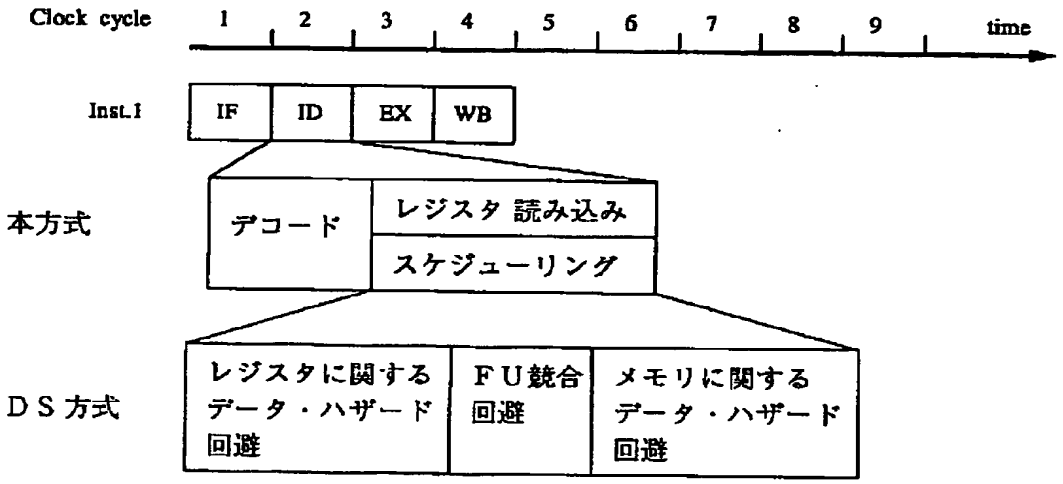
【図２】



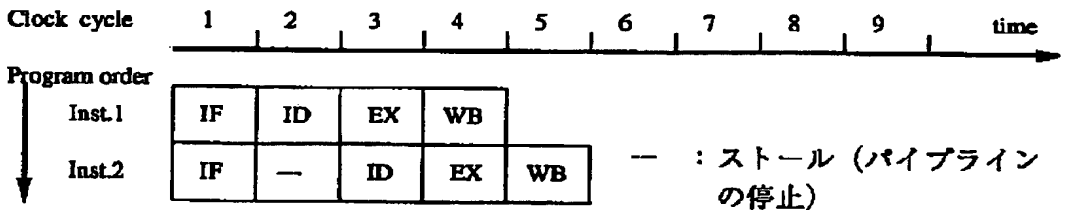
【図1】


バイパス
キャンセル
命令2～4が発行
可または不可
発行命令数
命令発行
可／不可
判別回路
40
30
32
バイパス
キャンセル
回路
メモリ
アドレス
競合検出
回路
33
スケジューリング手段
PLA
31
20
レジスタ
競合検出
回路
オペコード
オペランド
11 命令1
デコ
ーダ
1
12 命令2
デコ
ーダ
2
13 命令3
デコ
ーダ
3
14 命令4
デコ
ーダ
4

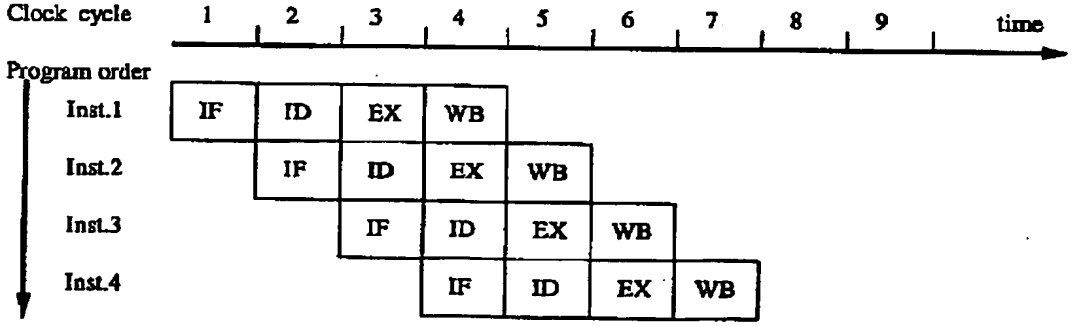
【図3】


Clock cycle
1
2
3
4
5
6
7
8
9
time
Inst.1
IF
ID
EX
WB
本方式
デコード
レジスタ 読み込み
スケジューリング
DS方式
レジスタに関するデータ・ハザード回避
FU競合回避
メモリに関するデータ・ハザード回避

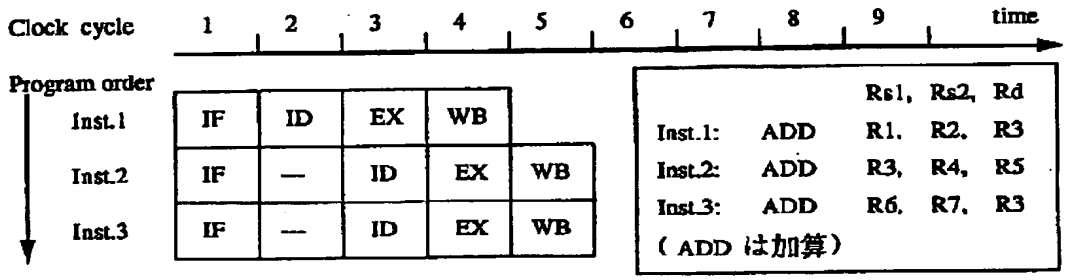
【図4】


Clock cycle
1
2
3
4
5
6
7
8
9
time
Program order
Inst.1
Inst.2
IF
ID
EX
WB
IF
—
ID
EX
WB
— ：ストール（パイプラインの停止）

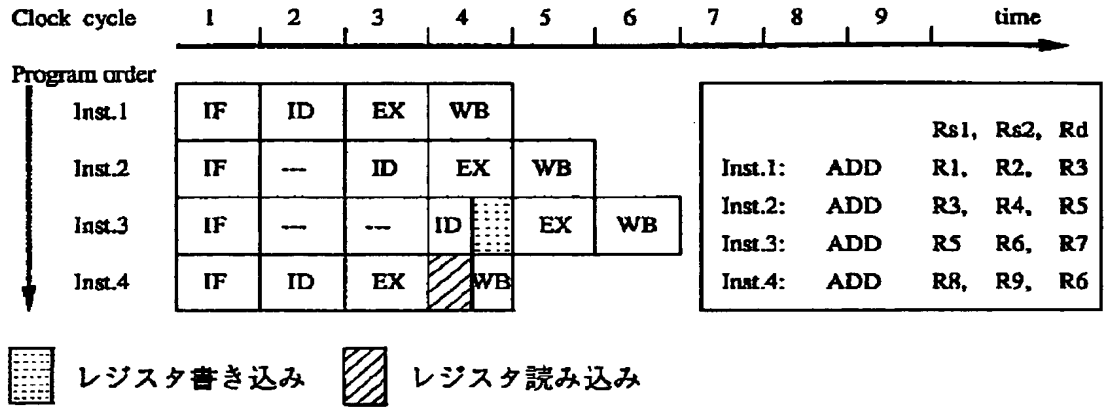
【図12】


Clock cycle
1
2
3
4
5
6
7
8
9
time
Program order
Inst.1
Inst.2
Inst.3
Inst.4
IF
ID
EX
WB

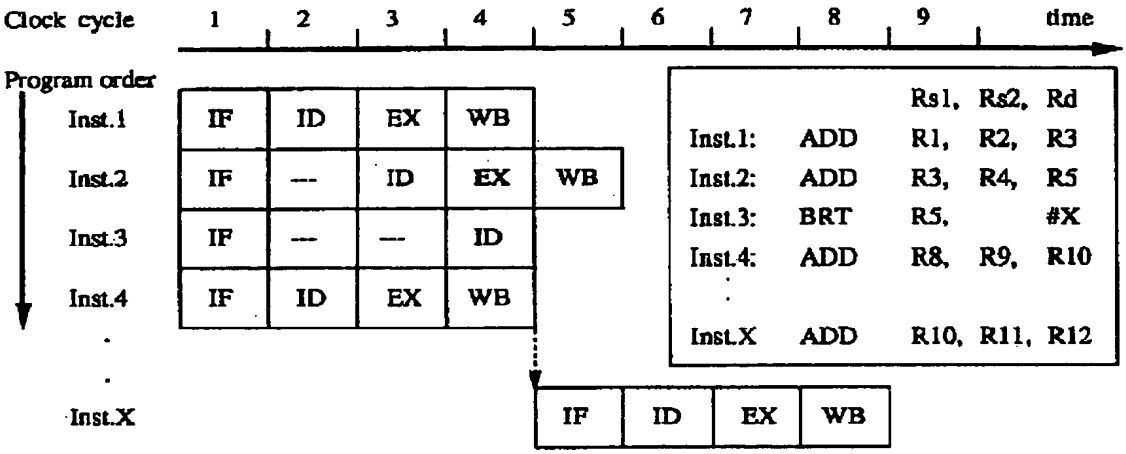
【図５】


Clock cycle
1
2
3
4
5
6
7
8
9
time
Program order
Inst.1
Inst.2
Inst.3
IF
ID
EX
WB
IF
—
ID
EX
WB
IF
—
ID
EX
WB
Rs1, Rs2, Rd
Inst.1: ADD R1, R2, R3
Inst.2: ADD R3, R4, R5
Inst.3: ADD R6, R7, R3
（ADD は加算）

【図６】

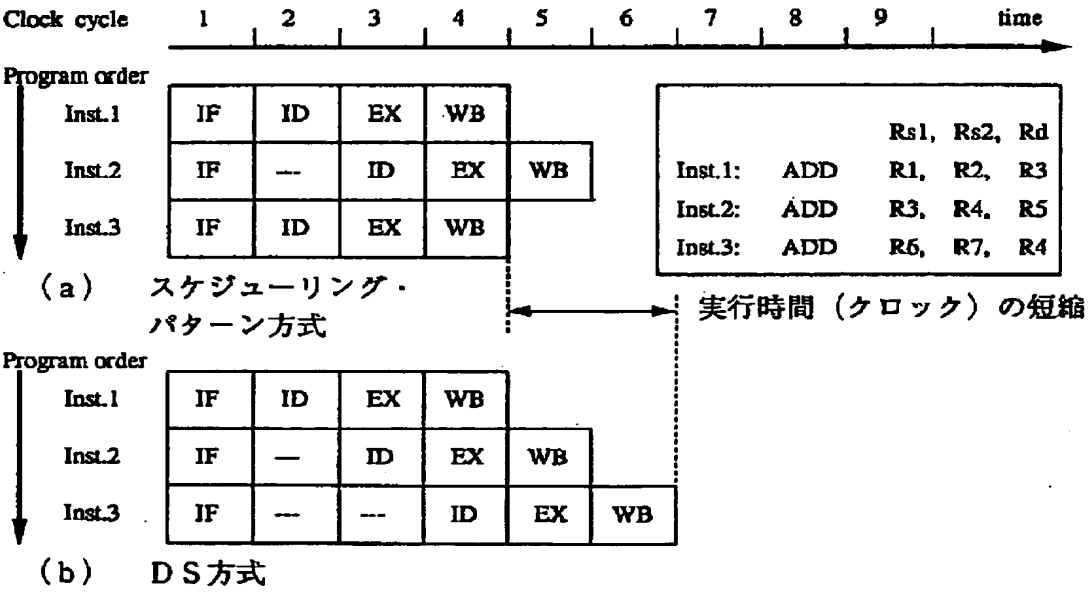

Clock cycle
1
2
3
4
5
6
7
8
9
time
Program order
Inst.1
Inst.2
Inst.3
Inst.4
IF
ID
EX
WB
IF
—
ID
EX
WB
IF
—
—
ID
EX
WB
IF
ID
EX
WB
Rs1, Rs2, Rd
Inst.1: ADD R1, R2, R3
Inst.2: ADD R3, R4, R5
Inst.3: ADD R5 R6, R7
Inst.4: ADD R8, R9, R6
レジスタ書き込み
レジスタ読み込み

【図7】


Clock cycle
1
2
3
4
5
6
7
8
9
time
Program order
Inst.1
Inst.2
Inst.3
Inst.4
Inst.X
IF
ID
EX
WB

Rs1, Rs2, Rd
Inst.1: ADD R1, R2, R3
Inst.2: ADD R3, R4, R5
Inst.3: BRT R5, #X
Inst.4: ADD R8, R9, R10
Inst.X ADD R10, R11, R12

【図8】

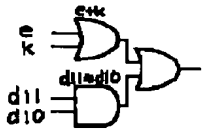

Clock cycle
1
2
3
4
5
6
7
8
9
time
Program order
Inst.1
Inst.2
Inst.3
IF
ID
EX
WB

Rs1, Rs2, Rd
Inst.1: ADD R1, R2, R3
Inst.2: ADD R3, R4, R5
Inst.3: ADD R6, R7, R4
（a） スケジューリング・パターン方式
実行時間（クロック）の短縮
Program order
Inst.1
Inst.2
Inst.3
IF
ID
EX
WB
—

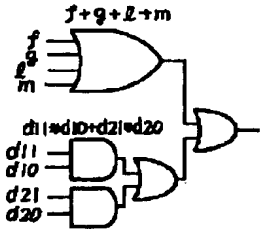
（b） ＤＳ方式

【図9】

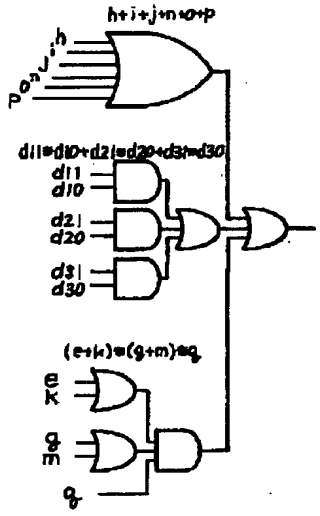
(a) 命令 2



(b) 命令 3

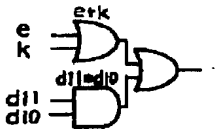


(c) 命令 4

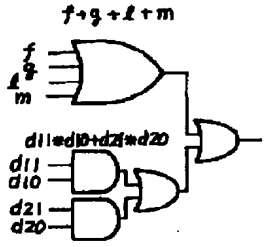


【図１０】

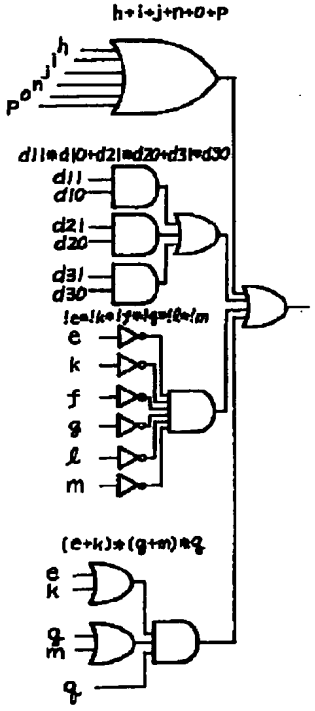
(a) 命令２



(b) 命令３

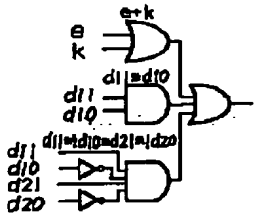


(c) 命令４

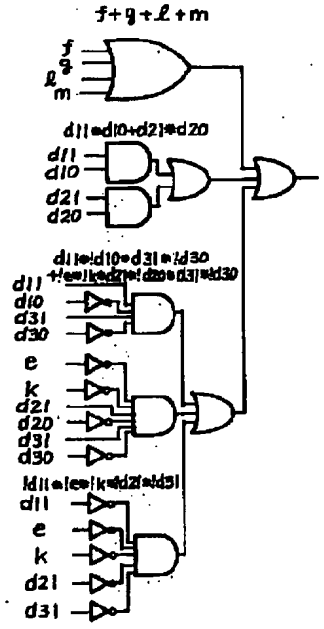


【図１１】

(a) 命令 2



(b) 命令 3



(c) 命令 4